Matrix Adaptation Evolution Strategies for Optimization Under Nonlinear Equality Constraints

Patrick Spettel[a,*], Hans-Georg Beyer[a]

[a]*Research Center Process and Product Engineering*
Vorarlberg University of Applied Sciences
Hochschulstr. 1, 6850 Dornbirn, Austria

Abstract

This work concerns the design of matrix adaptation evolution strategies for black-box optimization under nonlinear equality constraints. First, constraints in form of elliptical manifolds are considered. For those constraints, an algorithm is proposed that evolves itself on that manifold while optimizing the objective function. The specialty about the approach is that it is possible to ensure that the population evolves on the manifold with closed-form expressions. Second, an algorithm design for general nonlinear equality constraints is presented. For those constraints considered, an iterative repair approach is presented. This allows the evolution to happen on the nonlinear manifold defined by the equality constraints for this more general case as well. For both cases, the algorithms are interior point methods, i.e., the objective function is only evaluated at feasible points in the parameter space, which is often required in the area of simulation-based optimization. For the experimental evaluation, different test problems are introduced. The proposed algorithms are evaluated on those providing insights into the working principles of the different approaches. It is experimentally shown that correcting the mutation vectors after the repair step is important for an effective evolution strategy. Additional experiments are conducted for providing a comparison to other evolutionary black-box optimization methods, which show that the developed algorithms are competitive.

Keywords: Matrix adaptation evolution strategies, Nonlinear constraints, Nonlinear manifold, Experimental evaluation

1. Introduction

A variety of evolutionary algorithms (EAs) have been proposed for dealing with (black-box) optimization problems containing (black-box) constraints. An overview of different constraint handling approaches is for example given

[*]Corresponding author
Email addresses: Patrick.Spettel@fhv.at (Patrick Spettel), Hans-Georg.Beyer@fhv.at (Hans-Georg Beyer)

in [1]. Many algorithm designs are based on differential evolution (DE) [2] and incorporate different constraint handling ideas [3, 4, 5, 6]. Additionally, an evolution strategy (ES) incorporating the ϵ-level ordering and gradient-based repair from successful DE variants into the matrix adaptation evolution strategy (MA-ES) [7] has been proposed [8]. However, in contrast to e.g. DE methods, extensive study of constraint handling in ESs has only started. Therefore, our interest is to gain a better understanding of the possibilities of handling constraints in ESs. Since Covariance Matrix Adaptation (CMA) ESs such as the CMA-ES [9, 10] are arguably the most prominent variants, the idea is to incorporate constraint handling into such a variant. All of the methods mentioned above support a black-box objective function and black-box constraints. However, they evaluate parameter vectors outside of the feasible region. While there are problems for which this does not pose any issues, in the area of simulation-based optimization (SBO), the evaluation of infeasible search points is often not possible. Motivated by real-world applications like the multi-period portfolio optimization problem with transaction costs [11] or the financial stress-testing problem presented in [12], interior point methods are the aim of this work. In the latter work, random stress scenarios are considered and their quality is determined by a balance sheet simulation. The consideration of only plausible scenarios introduces an elliptic constraint. In addition, the search-space is bounded by box constraints, which is not considered in this paper.

In [13], an interior point ES was designed for optimization under linear constraints. It evolves itself on a linear manifold by projecting infeasible individuals onto the constraint manifold. The goal of this work is to extend that idea to nonlinear constraints. In contrast to the approach of discarding infeasible offspring, repair (by projection) is advantageous especially if the probability of sampling feasible individuals is small. A peculiarity of repair methods is that they influence the mutation vectors that are used for the covariance matrix (factor) and mutation strength updates. Hence, after the repair step, the mutation vectors are usually corrected (calculated back) such that the parent together with the mutation results again in the repaired offspring. Such a back-calculation has been performed in [13] for the linear constraints, however, its influence has not been investigated deeper. This work addresses that by experimentally evaluating methods with and without back-calculation on example problems with nonlinear constraints. The experimental results indicate that the back-calculation is to be preferred.

Besides the stress-testing problem, further application examples are Thomson's problem [14] and Tammes' problem [15]. The Thomson problem is the task of determining the position of a given number of electrons on a unit sphere in 3-dimensional space such that their electrostatic potential energy is minimized. Directly connected to that problem is Tammes' problem. Instead of optimizing an electrostatic potential energy, the objective is to maximize the minimum distance between any pair of a given number of points on the unit sphere in 3-dimensional space.

Considering theoretical aspects, an ES for optimization of a linear function on a spherical manifold has been theoretically investigated in [16].

Contributions

- Design and evaluation of interior point ESs for optimization under nonlinear constraints. Whereas in [13] only linear constraints are supported, the newly designed ESs in this work support nonlinear constraints. Furthermore, it is different than DE approaches such as [3, 4, 5, 6] since it ensures that the objective function is only evaluated for feasible search points, i.e., it is an interior point method.

- A repair approach is developed that supports nonlinear constraints. That is, infeasible offspring are not discarded. Related work in this direction has also been done by Arnold in [17, 18]. In those papers, repair is per-formed by solving an optimization problem for an infeasible search point that finds a feasible point that has minimal Euclidean distance to the in-feasible point. The developed ESs in this work are different: The first one considers an elliptical constraint that can be fulfilled by adapting the sampling step. The second one uses an iterative repair step based on gradients, whereas [17, 18] use Matlab's fmincon.
  Since the goal of this work is the design of interior point methods, repair is the chosen approach for the algorithm with the more general nonlin-ear equality constraints. However, fulfilling the nonlinear equality con-straints can be formulated as an optimization problem itself (e.g., projec-tion by minimizing the Euclidean distance). The proposed method with the gradient-based repair is arguably relatively simple and is shown to work well in the experiments.

The remainder of the paper is organized as follows: First, an approach for known nonlinear constraints defining elliptical manifolds is developed in Section 2. With the assumption of knowing the constraints for this case, an algorithm design is presented that enables the evolution on nonlinear manifolds with closed-form calculations. This is in contrast to the second algorithm that is presented in Section 3. There, general nonlinear equality constraints are considered. Hence, the repair is more complex and an iterative repair method is proposed. Experimental evaluation results of both the algorithms are presented in Section 4. Finally, the work is concluded with Section 5.

2. A Matrix Adaptation Evolution Strategy for Optimization on Elliptical Manifolds

In this section, optimization on elliptical manifolds is considered. The following two sections describe the problem (Section 2.1) and the algorithm (Section 2.2), respectively.

2.1. Problem Description

The intent is to start investigating extensions of [13], where linear constraints have been treated by evolution on linear manifolds. The consideration in this chapter is restricted to elliptical manifolds. As it turns out (see Section 2.2), the evolution on the manifolds considered here can be achieved by closed-form expressions. This is in contrast to the linear constraint handling in [13], where an iterative projection repair method was used.

The considered optimization problem can be formulated as

$$\begin{aligned} &\text{min. } f(\mathbf{x}) \\ &\text{s.t. } \mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa, \end{aligned} \tag{1}$$

where the objective function f can be a black-box function, $\mathbf{S} = \mathbf{S}^T$, and $\varkappa > 0$. That is, $\mathbf{S}$ is assumed to be symmetric and positive definite and hence $\mathbf{S} = \mathbf{A}^T\mathbf{A}$ can be obtained by a Cholesky decomposition. Problems of such form include the Thomson problem [14] and the Tammes problem [15]. They are explained in more detail in Section 4.1. The algorithm design is presented and explained in Section 2.2 and experimentally evaluated in Section 4.

2.2. Algorithm

Algorithm 1 shows the pseudo-code of the algorithm. It is the proposed $(\mu/\mu_w, \lambda)$-MA-ES algorithm from [7] with an adapted offspring generation and parental individual update ensuring that the offspring are created on the elliptical manifold and the parental individual for the next generation is also on the manifold. After the initialization (Lines 1 to 6), the generational loop is entered (Line 7) and it is run as long as the termination criteria are not fulfilled (Line 20). Each offspring is created in Lines 8 to 13 by sampling a standard normally distributed mutation vector $\tilde{\mathbf{z}}_l$ (Line 9). It is then multiplied with the covariance matrix factor $\mathbf{M}$ yielding $\tilde{\mathbf{d}}_l$ (Line 10). The creation of an offspring's parameter vector $\tilde{\mathbf{x}}_l \leftarrow \sqrt{\varkappa}\left(\frac{\mathbf{x}+\sigma\mathbf{A}^{-1}\tilde{\mathbf{d}}_l}{||\mathbf{A}\mathbf{x}+\sigma\tilde{\mathbf{d}}_l||}\right)$ in Line 11 is new and ensures that the created offspring are on the manifold (refer to Appendix A for the details concerning this update). Throughout the paper, $||\cdot||$ is used to denote the Euclidean norm $||\cdot||_2$. The fitness evaluation of an offspring finishes its creation (Line 12). After the offspring creation, they are sorted in ascending order according to their fitness (Line 14) and $\mathbf{x}$, $\mathbf{s}$, $\mathbf{M}$, and σ are updated (Lines 15 to 18). The updates of $\mathbf{s}$ and $\mathbf{M}$ are directly done as proposed in [7]. The update of σ is a simplified version and is analogous to [19]. Similar to the offspring creation, the update of the parental individual[1] for the next generation

$$\mathbf{x} \leftarrow \sqrt{\varkappa}\left(\frac{\mathbf{x}+\sigma\mathbf{A}^{-1}\sum_{m=1}^{\mu} w_m\tilde{\mathbf{d}}_{m;\lambda}}{||\mathbf{A}\mathbf{x}+\sigma\sum_{m=1}^{\mu} w_m\tilde{\mathbf{d}}_{m;\lambda}||}\right)$$

in Line 15 is new and ensures that it is on the manifold (again, refer to Appendix A for the details about this update). The update of the generation

[1]The notation $\mathbf{x}_{m;\lambda}$ is the order statistic notation and denotes the m-th best (w.r.t. fitness) out of λ values.

counter (Line 19) ends one iteration of the generational loop. Notice that since the objective function is not evaluated for the parental individual $\mathbf{x}$, the standard MA-ES update $\mathbf{x} \leftarrow \mathbf{x} + \sigma \sum_{m=1}^{\mu} w_m \tilde{\mathbf{d}}_{m;\lambda}$ can alternatively be used. However, in practical implementations one often keeps track of the best-so-far individual, where the recombinant is usually taken into account. In this situation, it is important that the recombinant satisfies the constraint $\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa$, which can be ensured by the expression in Line 15.

For deriving an upper bound of the asymptotic runtime of one generation of Algorithm 1, let T_f be an upper bound for the runtime of the objective function evaluation. Then, notice that the Cholesky decomposition $\mathbf{S} = \mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^{-1}$ can be precomputed as an initialization step in $O(N^3)$. Hence, for Lines 8 to 13 one obtains $O(\lambda N^2 T_f)$ (assuming $O(1)$ for the random number generation). For Lines 14 to 19 one gets $O(N^3)$ if Line 17 is implemented as a matrix-matrix multiplication. Note that it can also be implemented in $O(N^2)$ following an idea analogous to Equation (33). In total, for the asymptotic runtime of one generation one obtains $O(N^3)$ if λT_f scales at most linearly in N. If λT_f scales more than linearly in N, then one has $O(\lambda N^2 T_f)$.

Algorithm 1 The $(\mu/\mu_w, \lambda)$-MA-ES for optimization on elliptical manifolds $\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa$ in $\mathbb{R}^N$, where $\mathbf{S} = \mathbf{S}^T$, $\varkappa > 0$, and $\mathbf{S} = \mathbf{A}^T\mathbf{A}$ (by Cholesky decomposition).

```
1: Initialize parameters $\mu, \lambda, \mu_{\text{eff}}, c_s, c_1, c_w$, and weights $w_m$ for $1 \le m \le \mu$
2: Initialize $\mathbf{x}$
3: Initialize $\sigma$
4: $\mathbf{M} \leftarrow \mathbf{I}$
5: $\mathbf{s} \leftarrow \mathbf{0}$
6: $g \leftarrow 0$
7: repeat
8:     for $l \leftarrow 1$ to $\lambda$ do
9:         $\tilde{\mathbf{z}}_l \leftarrow \mathcal{N}_l(\mathbf{0}, \mathbf{I})$
10:        $\tilde{\mathbf{d}}_l \leftarrow \mathbf{M}\tilde{\mathbf{z}}_l$
11:        $\tilde{\mathbf{x}}_l \leftarrow \sqrt{\varkappa}\left(\frac{\mathbf{x}+\sigma\mathbf{A}^{-1}\tilde{\mathbf{d}}_l}{||\mathbf{A}\mathbf{x}+\sigma\tilde{\mathbf{d}}_l||}\right)$
12:        $\tilde{f}_l \leftarrow f(\tilde{\mathbf{x}}_l)$
13:    end for
14:    Sort offspring according to fitness in ascending order
15:    $\mathbf{x} \leftarrow \sqrt{\varkappa}\left(\frac{\mathbf{x}+\sigma\mathbf{A}^{-1}\sum_{m=1}^{\mu} w_m\tilde{\mathbf{d}}_{m;\lambda}}{||\mathbf{A}\mathbf{x}+\sigma\sum_{m=1}^{\mu} w_m\tilde{\mathbf{d}}_{m;\lambda}||}\right)$
16:    $\mathbf{s} \leftarrow (1-c_s)\,\mathbf{s} + \sqrt{\mu_{\text{eff}}c_s\,(2-c_s)}\sum_{m=1}^{\mu} w_m\tilde{\mathbf{z}}_{m;\lambda}$
17:    $\mathbf{M} \leftarrow \mathbf{M}\left[\mathbf{I} + \frac{c_1}{2}\left(\mathbf{s}\mathbf{s}^T - \mathbf{I}\right) + \frac{c_w}{2}\left(\left(\sum_{m=1}^{\mu} w_m\tilde{\mathbf{z}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^T\right) - \mathbf{I}\right)\right]$
18:    $\sigma \leftarrow \sigma \exp\left[\frac{c_s}{2}\left(\frac{||\mathbf{s}||^2}{N} - 1\right)\right]$
19:    $g \leftarrow g + 1$
20: until termination criteria fulfilled
```

3. A Matrix Adaptation Evolution Strategy for Optimization Under General Nonlinear Equality Constraints

The topic of this section is the optimization considering more general nonlinear equality constraints. Section 3.1 describes the problem and Section 3.2 presents the algorithm.

3.1. General Problem Formulation

Section 2 considered the evolution on elliptical manifolds. This chapter investigates an extension to more general nonlinear equality constraints. The considered optimization problem can be stated as

$$\begin{aligned}&\text{min. } f(\mathbf{x})\\ &\text{s.t. } h_k(\mathbf{x}) = 0 \text{ for } k \in \{1, \ldots, K\},\end{aligned} \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^N$ and K is the number of constraints. The goal is again to design an interior point method. That is, the evolution should happen on the manifold defined by the nonlinear equality constraints. Since the objective function and the constraint functions can be black-box functions, no assumptions are made about their form. For treating infeasible individuals, a general repair approach is followed. The algorithm with its repair approach is presented in Section 3.2 and experimentally evaluated in Section 4.

3.2. Algorithm

Algorithm 2 shows the pseudo-code of the algorithm. After the initialization in Lines 1 to 7, the generational loop is entered (Line 8) and it is run as long as the termination criteria are not fulfilled (Line 38). In every generation, λ offspring are created (Lines 9 to 21): Each offspring is created by sampling a standard normally distributed mutation vector $\tilde{\mathbf{z}}_l$, multiplying it with the covariance matrix factor $\mathbf{M}$, scaling it with the parental mutation strength σ, and adding the result to the parental individual's parameter vector $\mathbf{x}$. If the resulting offspring's parameter vector is not feasible, it is repaired and the mutation vectors are back-calculated if the configuration is set as such. The evaluation of the offspring's fitness ends the offspring creation loop. After that, the offspring are sorted in ascending order (Line 22). The μ best offspring are used to update $\mathbf{x}$ (Line 23), which is again repaired if it results in an infeasible parameter vector (Lines 24 to 26). After that, $\mathbf{s}$, $\mathbf{M}$, $\mathbf{M}_{\text{inv}}$, σ, and the generation counter g are updated (Lines 27 to 37). The updates of $\mathbf{s}$ and $\mathbf{M}$ are directly done as proposed in [7]. The update of σ is a simplified version and is analogous to [19]. The iterative update of $\mathbf{M}_{\text{inv}}$ is explained in Section 3.2.3. It is derived similar to the update of $\mathbf{M}$. The functions *shouldDoBackCalculation* and *shouldDoIterative-UpdateOfMInverse* are Boolean functions. They are used to indicate different variants of the same algorithm. The former indicates whether an offspring's mutation vector should be calculated back after repair. The latter is used to differentiate between two different ways for the $\mathbf{M}_{\text{inv}}$ computation (either an iterative update from one generation to the next or always a full inverse computation[2]). The different variants are experimentally evaluated in Section 4. The main intention of the back-calculation is to have a mutation vector that results in the offspring after repair, since it influences the learning of the covariance matrix factor.

For deriving an upper bound of the asymptotic runtime of one generation of Algorithm 2, let T_f be again the upper bound for the runtime of the objective function evaluation and let T_c be the upper bound for the evaluation of the constraints. Additionally, let T_r be an upper bound for the repair algorithm (Algorithm 3). Since the computation of the pseudo-inverse dominates one iteration of Algorithm 3 and the maximum number of iterations is bounded above by T, one has $T_r = O(T\,N^3 T_c)$ for the case involving the computation of the inverse of the Jacobian and $T_r = O(T\,N T_c)$ for the special case of one equality constraint (Section 3.2.2). Using those observations, for Lines 9 to 21 one obtains $O(\lambda N^2 T_f T_c T_r)$ (assuming $O(1)$ for the random number generation). For Lines 14 to 19 one gets $O(N^3)$ considering all possible branches. In total, for the asymptotic runtime of one generation one obtains $O(N^3)$ if $\lambda T_f\, T_c T_r$ scales at most linearly in N. If $\lambda T_f\, T_c T_r$ scales more than linearly in N, then one has $O(\lambda N^2 T_f T_c T_r)$.

3.2.1. Repair – General Case

Assume the N-dimensional problem with K equality constraints $h_k(\mathbf{x}) = 0$ for $k \in \{1, \dots, K\}$ presented as Equation (2). The goal is to repair a vector $\mathbf{x}$ that violates some of those K constraints. More formally, given a vector $\mathbf{x}$ with

$$h_k(\mathbf{x}) = d_k, \tag{3}$$

where d_k denotes the error of $\mathbf{x}$ for the the k-th constraint, the goal is to find $\mathbf{\Delta x}$ such that

$$h_k(\mathbf{x} + \mathbf{\Delta x}) \overset{!}{=} 0 \tag{4}$$

for all $k \in \{1, \dots, K\}$. Note that if for all $k \in \{1, \dots, K\}$ it is true that $d_k = 0$, $\mathbf{x}$ is feasible and no repair is necessary.

By applying a Taylor expansion with cut-off after the linear term to the left-hand side of Equation (4), one obtains[3]

$$h_k(\mathbf{x}) + \sum_{i=1}^{N} \frac{\partial\, h_k}{\partial\,(\mathbf{x})_i} (\mathbf{\Delta x})_i \overset{!}{=} 0. \tag{5}$$

[2]In practical implementations, one might consider using the pseudoinverse. It makes the inverse less exact but can handle covariance matrix factors that are degenerate to some extent.

[3]Note that x_i and $(\mathbf{x})_i$ are equivalent notations used for indicating the i-th element of a vector $\mathbf{x}$.

Algorithm 2 The $(\mu/\mu_w, \lambda)$-MA-ES for optimization under general nonlinear equality constraints in $\mathbb{R}^N$.

```
1: Initialize parameters $\mu, \lambda, \mu_{\text{eff}}, c_s, c_1, c_w$, and weights $w_m$ for $1 \leq m \leq \mu$
2: Initialize $\mathbf{x}$
3: Initialize $\sigma$
4: $\mathbf{M} \leftarrow \mathbf{I}$
5: $\mathbf{M}_{\text{inv}} \leftarrow \mathbf{I}$
6: $\mathbf{s} \leftarrow \mathbf{0}$
7: $g \leftarrow 0$
8: repeat
9:     for $l \leftarrow 1$ to $\lambda$ do
10:        $\tilde{\mathbf{z}}_l \leftarrow \mathcal{N}_l(\mathbf{0}, \mathbf{I})$
11:        $\tilde{\mathbf{d}}_l \leftarrow \mathbf{M}\tilde{\mathbf{z}}_l$
12:        $\tilde{\mathbf{x}}_l \leftarrow \mathbf{x} + \sigma\tilde{\mathbf{d}}_l$
13:        if not isFeasible($\tilde{\mathbf{x}}_l$) then
14:            $\tilde{\mathbf{x}}_l \leftarrow$ repair($\tilde{\mathbf{x}}_l$)                ▷ see Algorithm 3
15:            if shouldDoBackCalculation() then
16:                $\tilde{\mathbf{d}}_l \leftarrow \frac{\tilde{\mathbf{x}}_l - \mathbf{x}}{\sigma}$
17:                $\tilde{\mathbf{z}}_l \leftarrow \mathbf{M}_{\text{inv}}\tilde{\mathbf{d}}_l$
18:            end if
19:        end if
20:        $\tilde{f}_l \leftarrow f(\tilde{\mathbf{x}}_l)$
21:    end for
22:    Sort offspring according to fitness in ascending order
23:    $\mathbf{x} \leftarrow \mathbf{x} + \sigma \sum_{m=1}^{\mu} w_m \tilde{\mathbf{d}}_{m;\lambda}$
24:    if not isFeasible($\mathbf{x}$) then
25:        $\mathbf{x} \leftarrow$ repair($\mathbf{x}$)                ▷ see Algorithm 3
26:    end if
27:    $\mathbf{s} \leftarrow (1 - c_s)\,\mathbf{s} + \sqrt{\mu_{\text{eff}} c_s\,(2 - c_s)} \sum_{m=1}^{\mu} w_m \tilde{\mathbf{z}}_{m;\lambda}$
28:    $\mathbf{M} \leftarrow \mathbf{M}\left[\mathbf{I} + \frac{c_1}{2}\left(\mathbf{s}\mathbf{s}^T - \mathbf{I}\right) + \frac{c_w}{2}\left(\left(\sum_{m=1}^{\mu} w_m \tilde{\mathbf{z}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^T\right) - \mathbf{I}\right)\right]$
29:    if shouldDoBackCalculation() then
30:        if shouldDoIterativeUpdateOfMInverse() then
31:            $\mathbf{M}_{\text{inv}} \leftarrow \left[\mathbf{I} - \frac{c_1}{2}\left(\mathbf{s}\mathbf{s}^T - \mathbf{I}\right) - \frac{c_w}{2}\left(\left(\sum_{m=1}^{\mu} w_m \tilde{\mathbf{z}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^T\right) - \mathbf{I}\right)\right]\mathbf{M}_{\text{inv}}$
32:        else
33:            $\mathbf{M}_{\text{inv}} \leftarrow \mathbf{M}^{-1}$
34:        end if
35:    end if
36:    $\sigma \leftarrow \sigma \exp\left[\frac{c_s}{2}\left(\frac{||\mathbf{s}||^2}{N} - 1\right)\right]$
37:    $g \leftarrow g + 1$
38: until termination criteria fulfilled
```

By insertion of Equation (3) into Equation (5), one can write

$$\sum_{i=1}^{N} \frac{\partial h_k}{\partial (\mathbf{x})_i} (\mathbf{\Delta x})_i = -d_k. \tag{6}$$

Studying Equation (6), one notes that $\frac{\partial h_k}{\partial (\mathbf{x})_i}$ is exactly the entry of the Jacobi matrix $\mathbf{J}$ on the k-th row and i-th column. Hence, one can write

$$\sum_{i=1}^{N} (\mathbf{J})_{ki} (\mathbf{\Delta x})_i = -d_k, \tag{7}$$

which can be written in matrix-vector form as

$$\mathbf{J}\mathbf{\Delta x} = -\mathbf{d}, \tag{8}$$

where $\mathbf{J} \in \mathbb{R}^{K\times N}$, $\mathbf{\Delta x} \in \mathbb{R}^N$, and $\mathbf{d} \in \mathbb{R}^K$. $\mathbf{J}$ in Equation (8) is not invertible in general. Consequently, an alternative way of solving Equation (8) for $\mathbf{\Delta x}$ is to compute a least-squares solution. To this end, multiplying both sides of Equation (8) with $\mathbf{J}^T$ from the left yields

$$\mathbf{J}^T\mathbf{J}\mathbf{\Delta x} = -\mathbf{J}^T\mathbf{d}. \tag{9}$$

Provided that $\mathbf{J}^T\mathbf{J}$ has full rank N, it is invertible and one gets

$$\mathbf{\Delta x} = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{d}. \tag{10}$$

Otherwise, if $\mathbf{J}^T\mathbf{J}$ is not invertible, one can use Tikhonov's regularization with a regularizer α for the smallest $||\mathbf{\Delta x}||$:

$$\mathbf{\Delta x} = -\lim_{\alpha\to 0} \left[(\mathbf{J}^T\mathbf{J} + \alpha\mathbf{I})^{-1}\mathbf{J}^T\right]\mathbf{d}. \tag{11}$$

Since the limit expression in Equation (10) is a definition of the pseudo-inverse $\mathbf{J}^+$ (refer for example to [20] for more details), one can write

$$\mathbf{\Delta x} = -\mathbf{J}^+\mathbf{d}. \tag{12}$$

Since the quadratic and higher-order terms of the Taylor expansion have been neglected from Equation (4) to Equation (5), Equation (12) inserted into the condition stated as Equation (4) does usually not satisfy that condition. Nevertheless, the derived result in Equation (12) can be used as an iterative improvement method as depicted in Algorithm 3. Note that depending on how ϵ is set, a maximum number of iterations can be used as an additional termination criterion in practical implementations of the algorithm. Since the constraint functions h_k can be black-box functions, a limit for the maximum number of repair iterations also ensures that the loop is quit eventually in case that it does not converge.

One way to numerically compute the entries of the Jacobi matrix in Line 6 of Algorithm 3 is by using the central difference method. More formally,

$$\frac{\partial\, h_k}{\partial\, (\mathbf{x}^{(t)})_i} \approx \frac{h_k(\mathbf{x}^{(t)} + \mathbf{\Xi}_i) - h_k(\mathbf{x}^{(t)} - \mathbf{\Xi}_i)}{2\xi},$$

where

$$\xi = \begin{cases} \epsilon_J & \text{if } (\mathbf{x}^{(t)})_i = 0 \\ |(\mathbf{x}^{(t)})_i|\, \epsilon_J & \text{otherwise} \end{cases}$$

with a small $\epsilon_J > 0$. The notation $\mathbf{\Xi}_i$ indicates the i-th column vector of

$$\mathbf{\Xi} = \xi\, \mathbf{I}^{N\times N}.$$

Algorithm 3 The iterative repair method based on gradients for K equality constraints $h_k(\mathbf{x})$ with $k \in \{1,\ldots,K\}$. Note that those K functions can also be written as $\mathbf{h}(\mathbf{x})$ returning a K-dimensional vector $\mathbf{d} \in \mathbb{R}^K$.

```
1:  function repair($\mathbf{x}$)
2:      Initialize $\epsilon$
3:      $\mathbf{x}^{(0)} \leftarrow \mathbf{x}$
4:      $t \leftarrow 0$
5:      repeat
6:          Compute (numerically) the Jacobian $\mathbf{J}^{(t)}$, where $(\mathbf{J}^{(t)})_{ki} = \frac{\partial\, h_k}{\partial\, (\mathbf{x}^{(t)})_i}$
7:          $\mathbf{J}^{+} \leftarrow$ pseudo-inverse of $\mathbf{J}$
8:          $\mathbf{d}^{(t)} \leftarrow \mathbf{h}(\mathbf{x}^{(t)})$
9:          $\mathbf{x}^{(t+1)} \leftarrow \mathbf{x}^{(t)} - \mathbf{J}^{+}\mathbf{d}^{(t)}$
10:         $t \leftarrow t + 1$
11:     until $||\mathbf{h}(\mathbf{x}^{(t)})|| < \epsilon$ or $t \geq T$
12:     return $(\mathbf{x}^{(t)})$
13: end function
```

3.2.2. Repair – Special Case of One Equality Constraint

Considering an N-dimensional problem with only one (i.e., $K = 1$) equality constraint $h(\mathbf{x}) = 0$, one can derive an iterative repair method making use of the gradient ∇h as a special case from the general case theory presented above. With $K = 1$, the Jacobian writes

$$\mathbf{J} = \left(\frac{\partial\, h}{\partial\, x_1}, \ldots, \frac{\partial\, h}{\partial\, x_N}\right) = \nabla^T h. \tag{13}$$

From Equation (11), one has

$$\mathbf{\Delta x} = -\lim_{\alpha\to 0}\left[\left(\nabla h \nabla^T h + \alpha \mathbf{I}\right)^{-1} \nabla h\right] d. \tag{14}$$

For computing the inverse $\left(\nabla h \nabla^T h + \alpha \mathbf{I}\right)^{-1}$, a spectral decomposition

$$\nabla h \nabla^T h + \alpha \mathbf{I} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T \tag{15}$$

is performed, where $\mathbf{U}$ is an orthonormal matrix with the eigenvectors as columns ($\mathbf{U}^T \mathbf{U} = \mathbf{I}$) and $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues. Since $\mathbf{U}$ is an orthonormal matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix, the inverse can be obtained by

$$\left(\mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^T\right)^{-1} = \mathbf{U} \boldsymbol{\Lambda}^{-1} \mathbf{U}^T, \tag{16}$$

where

$$\left(\boldsymbol{\Lambda}^{-1}\right)_{ij} = \begin{cases} 1/(\boldsymbol{\Lambda})_{ij} & \text{if } i = j \\ (\boldsymbol{\Lambda})_{ij} = 0 & \text{otherwise.} \end{cases} \tag{17}$$

The calculation of the spectral decomposition leads to the eigenvalue problem

$$\left(\nabla h \nabla^T h + \alpha \mathbf{I}\right) \mathbf{u}_m = \lambda_m \mathbf{u}_m. \tag{18}$$

For the first normalized eigenvector, one gets

$$\mathbf{u}_1 = \frac{\nabla h}{||\nabla h||}. \tag{19}$$

Insertion of Equation (19) into Equation (18) leads to

$$\nabla h ||\nabla h|| + \frac{\alpha}{||\nabla h||} \nabla h = \lambda_1 \frac{\nabla h}{||\nabla h||}, \tag{20}$$

which implies

$$\lambda_1 = ||\nabla h||^2 + \alpha. \tag{21}$$

Since the intention is to have orthonormal eigenvectors and the first eigenvector (Equation (19)) is in direction of ∇h, one has

$$\mathbf{u}_m^T \nabla h = 0, \mathbf{u}_m^T \mathbf{u}_{m'} = \delta_{mm'} \text{ for } m \in \{2, \ldots, N\}, \tag{22}$$

where

$$\delta_{ij} = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases} \tag{23}$$

is the Kronecker delta function. Consideration of Equation (22) for calculating the m-th eigenvalue using Equation (18) leads to

$$\lambda_m = \alpha \text{ for } m \in \{2, \ldots, N\}. \tag{24}$$

Usage of Equation (19) and Equation (22) with Equation (15) results in

$$\left(\nabla h \nabla^T h + \alpha \mathbf{I}\right) = \left(||\nabla h||^2 + \alpha\right) \frac{\nabla h \nabla^T h}{||\nabla h||^2} + \alpha \sum_{m=2}^{N} \mathbf{u}_m \mathbf{u}_m^T. \tag{25}$$

For the inverse, notice that one only needs to invert the diagonal matrix $\mathbf{\Lambda}$ of eigenvalues as shown in Equations (16) and (17), which leads to

$$\left(\nabla h \nabla^T h + \alpha \mathbf{I}\right)^{-1} = \frac{1}{(||\nabla h||^2 + \alpha)} \frac{\nabla h \nabla^T h}{||\nabla h||^2} + \frac{1}{\alpha} \sum_{m=2}^{N} \mathbf{u}_m \mathbf{u}_m^T. \qquad (26)$$

Insertion of Equation (26) into Equation (14) yields

$$\begin{aligned}\mathbf{\Delta x} &= -\lim_{\alpha \to 0} \left[\frac{1}{(||\nabla h||^2 + \alpha)} \frac{\nabla h \overbrace{(\nabla^T h \nabla h)}^{=||\nabla h||^2}}{||\nabla h||^2} + \frac{1}{\alpha} \sum_{m=2}^{N} \mathbf{u}_m \underbrace{(\mathbf{u}_m^T \nabla h)}_{=0} \right] d \\ &= \left[\frac{\nabla h}{(||\nabla h||^2 + \alpha)} \right] d \\ &= -\frac{d}{||\nabla h||^2} \nabla h. \qquad (27)\end{aligned}$$

Equation (27) can now be used to design an iterative repair algorithm for $K = 1$ similar to Algorithm 3. In particular, Equation (27) replaces the expression involving the pseudo-inverse in Line 9 of Algorithm 3. Again, the entries of the gradient can be numerically computed by the central difference method.

3.2.3. Iterative Update of $\mathbf{M}_{inv}$

The motivation for an iterative update of $\mathbf{M}_{\text{inv}}$ is two-fold. First, inversion of $\mathbf{M}$ can lead to numerical problems in the case that it degenerates to some extent. Second, as shown below, it can be implemented as matrix-vector operations requiring less computation time.

Similar to the derivation leading to the update of $\mathbf{M}$ in Line 28 of Algorithm 2, the update of $\mathbf{M}_{\text{inv}}$ in Line 31 of Algorithm 2 can be derived. The derivation is briefly summarized here, but for the details regarding the $\mathbf{M}$-update it is referred to the derivations leading to [7, Eq. (30)]. The starting point is

$$\mathbf{M}^{(g+1)} = \mathbf{M}^{(g)} \left[\mathbf{I} + \mathbf{B}^{(g)}\right], \qquad (28)$$

where

$$\mathbf{B}^{(g)} = \frac{c_1}{2} \left(\mathbf{s}\mathbf{s}^T - \mathbf{I}\right) + \frac{c_w}{2} \left(\left(\sum_{m=1}^{\mu} w_m \tilde{\mathbf{z}}_{m;\lambda} \tilde{\mathbf{z}}_{m;\lambda}^T \right) - \mathbf{I} \right). \qquad (29)$$

For the inverse of $\mathbf{M}^{(g)}$, it holds

$$\begin{aligned}\left(\mathbf{M}^{(g+1)}\right)^{-1} &= \left(\mathbf{M}^{(g)} \left[\mathbf{I} + \mathbf{B}^{(g)}\right]\right)^{-1} \\ &= \left(\mathbf{I} + \mathbf{B}^{(g)}\right)^{-1} \left(\mathbf{M}^{(g)}\right)^{-1}. \qquad (30)\end{aligned}$$

Now, $\left(\mathbf{I}+\mathbf{B}^{(g)}\right)^{-1}$ can be expanded by a matrix power series into

$$\left(\mathbf{I}+\mathbf{B}^{(g)}\right)^{-1}=\mathbf{I}-\mathbf{B}^{(g)}+\left(\mathbf{B}^{(g)}\right)^{2}-\cdots. \tag{31}$$

Neglecting the quadratic and higher order terms, one gets

$$\begin{aligned}\left(\mathbf{M}^{(g+1)}\right)^{-1}&=\left[\mathbf{I}-\mathbf{B}^{(g)}\right]\left(\mathbf{M}^{(g)}\right)^{-1}\\&=\left[\mathbf{I}-\frac{c_1}{2}\left(\mathbf{s}\mathbf{s}^T-\mathbf{I}\right)-\frac{c_w}{2}\left(\left(\sum_{m=1}^{\mu}w_m\tilde{\mathbf{z}}_{m;\lambda}\tilde{\mathbf{z}}_{m;\lambda}^T\right)-\mathbf{I}\right)\right]\left(\mathbf{M}^{(g)}\right)^{-1}.\end{aligned} \tag{32}$$

If one implements Equation (32) as it is written, it is a matrix-matrix multiplication. One can reduce the computation time by implementing a rewritten form that results in matrix-vector multiplications. This idea is analogous to the one proposed in [19] for the $\mathbf{M}$ update: By distributing $\left(\mathbf{M}^{(g)}\right)^{-1}$ over the terms in the square bracket, one obtains

$$\begin{aligned}\left(\mathbf{M}^{(g+1)}\right)^{-1}=\left(1+\frac{c_1}{2}+\frac{c_w}{2}\right)\left(\mathbf{M}^{(g)}\right)^{-1}-\frac{c_1}{2}\mathbf{s}\left(\mathbf{s}^T\left(\mathbf{M}^{(g)}\right)^{-1}\right)\\-\frac{c_w}{2}\sum_{m=1}^{\mu}w_m\tilde{\mathbf{z}}_{m;\lambda}\left(\tilde{\mathbf{z}}_{m;\lambda}^T\left(\mathbf{M}^{(g)}\right)^{-1}\right).\end{aligned} \tag{33}$$

Hence, one can reduce the update of $\mathbf{M}$ and $\mathbf{M}_{\text{inv}}$ to $\Theta(N^2)$.

4. Experimental Evaluation

Thomson's problem [14] is a problem for which both presented algorithms (Algorithm 1 and Algorithm 2) can be experimentally evaluated and compared because it is an optimization problem constrained to the surface of a sphere. Directly related to Thomson's problem is Tammes' problem [15]. Both problems differ only w.r.t. their objective functions as described in Section 4.1.

Another interesting problem for the experimental evaluation of Section 3.2 in comparison to other black-box optimization methods that support nonlinear equality constraints is the problem of maximizing the area of a polygon constrained to having a given circumference. A more detailed description is given in Section 4.2.

4.1. Thomson's Problem

Thomson's problem is the problem of determining the position of M electrons on a unit sphere in 3-dimensional space such that their electrostatic potential energy is minimized. More formally, an abstract version of the problem can be

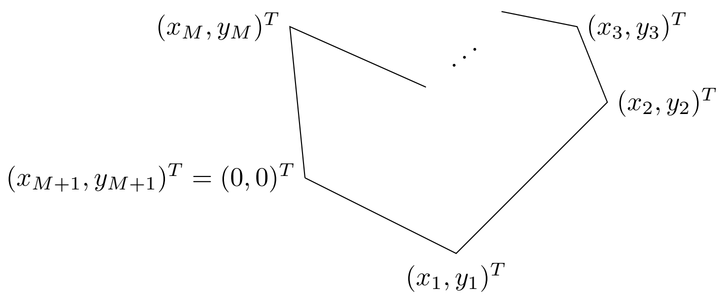

Figure 1: Visualization of a polygon in $\mathbb{R}^2$ with $M+1$ nodes numbered in counterclockwise order.

stated as

$$\begin{aligned} &\min \sum_{i=2}^{M} \sum_{j=1}^{i-1} \frac{1}{||\mathbf{r}_i - \mathbf{r}_j||} \\ &\text{s.t. } \forall k \in \{1, \dots, M\} : ||\mathbf{r}_k|| = 1, \end{aligned} \tag{34}$$

where $\mathbf{r}_i = (x_i, y_i, z_i)^T$. That is, $\mathbf{r}_i$ is a 3-dimensional vector containing the x, y, and z coordinates of the corresponding electron.

Tammes' problem is directly connected to Thomson's problem and results in the same optimal values for some values of M. Instead of optimizing an electrostatic potential energy, the objective is to maximize the minimum distance between any pair of M points on the unit sphere:

$$\begin{aligned} &\max \min_{i,j} ||\mathbf{r}_i - \mathbf{r}_j|| \\ &\text{s.t. } \forall k \in \{1, \dots, M\} : ||\mathbf{r}_k|| = 1, \end{aligned} \tag{35}$$

where $\mathbf{r}_i = (x_i, y_i, z_i)^T$. Again, $\mathbf{r}_i$ is a 3-dimensional vector containing the x, y, and z coordinates of the corresponding point.

4.2. Polygon With Fixed Perimeter Constraint

Given a number of points M and a circumference L, the goal of the polygon problem is to optimize the position of $M+1$ nodes in the 2-dimensional space $\mathbb{R}^2$ such that the circumference is L and the area is maximal. It is assumed w.l.o.g. that the $M+1$-st node is at the origin $(0,0)^T$. Let x_i and y_i denote the x coordinate and the y coordinate, respectively, of the i-th point. A visualization is given in Figure 1, in which a polygon with $M+1$ nodes in counterclockwise order is shown in the $\mathbb{R}^2$ space.

More formally, the optimization problem can be stated as

$$\begin{aligned} &\text{min. } A_{\max} - A(\mathbf{x}, \mathbf{y}) \\ &\text{s.t. } h(\mathbf{x}, \mathbf{y}) = 0, \end{aligned} \tag{36}$$

where $A_{\max}$ is the maximal possible area[4], $A(\mathbf{x}, \mathbf{y})$ is the area of the polygon defined by the x and y coordinates of all the points given as the vectors $\mathbf{x}$ and $\mathbf{y}$, respectively, and $h(\mathbf{x}, \mathbf{y})$ represents the equality constraint. The $A_{\max}$ value can be derived as

$$A_{\max} = \frac{L^2}{4 \tan \frac{\pi}{M+1}} \frac{1}{M+1}. \tag{37}$$

For the derivation of this formula, it is referred to Appendix B.

Since non-self-intersecting (i.e., simple) polygons are considered, the (signed) area $A(\mathbf{x}, \mathbf{y})$ can be calculated using the so-called shoelace formula as

$$A(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_{i=1}^{M-1} (x_i y_{i+1} - x_{i+1} y_i). \tag{38}$$

It yields a positive area if the points of the polygon are in counterclockwise order. In addition, remember again that $(x_{M+1}, y_{M+1})^T = (0, 0)^T$ and hence the products involving the $M + 1$-st node yield 0.

The equality constraint can be calculated as the sum of all the segments of the polygon. Remembering that $(x_{M+1}, y_{M+1})^T = (0, 0)^T$, it can be written using the Pythagorean Theorem as

$$h(\mathbf{x}, \mathbf{y}) = \sqrt{x_1^2 + y_1^2} + \left(\sum_{i=1}^{M} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \right) - L. \tag{39}$$

4.3. Results

For the experimental evaluation, Thomson's problem, the polygon problem, and problems with only nonlinear equality constraints from the CEC 2006 competition[5] on constrained optimization [23] have been implemented in the COCO framework [24]. The code is provided in a GitHub fork of the COCO framework, `https://github.com/patsp/coco`. The changes are in the new branch `development-sppa-manifold`. This branch is based on the `development` branch of `https://github.com/numbbo/coco`. The new code for the Thomson problem, the polygon problem, and the CEC 2006 problems with only nonlinear constraints[6] extends the test suite called `custom` (`code-experiments/src/suite_custom.c`, `code-experiments/src/coco_suite.c`).

[4]$A_{\max}$ is used to have the minimum at 0. This allows for a better assessment of the optimization dynamic. However, it is not needed for the algorithm to work.

[5]Further test problems of the CEC 2010 and CEC 2017 competitions on constrained optimization [21, 22] with only nonlinear equality constraints have been considered. However, note that in order for the benchmark to be the same as for the Thomson and the polygon problem, the optima of the problems have to be known to define targets. Hence, only the CEC 2006 problems have been used in this work.

[6]Notice that box constraints are not considered in our proposed method, however, the CEC 2006 g17 problem requires them. Hence, that problem's objective function has been adapted to return a large value for search points that are outside of the box constraints.

The implementations of those problems exhibit two peculiarities that are explained in the following two paragraphs.

Following the convention of the `bbob-constrained` COCO suite, which only supports inequality constraints, each equality constraint in the implementation of the Thomson problem and the polygon problem is implemented using two inequality constraints with an error tolerance of 10^{-8}. That is, a particular equality constraint $h_k(\mathbf{x}) = 0$ is represented as $-10^{-8} \leq h_k(\mathbf{x}) \leq 10^{-8}$. More explicitly, one can introduce two inequality constraints $g_k(\mathbf{x}) \leq 0$ and $g'_k(\mathbf{x}) \leq 0$ for each equality constraint $h_k(\mathbf{x})$, where $g_k(\mathbf{x}) := -h_k(\mathbf{x}) - 10^{-8} \leq 0$ and $g'_k(\mathbf{x}) := h_k(\mathbf{x}) - 10^{-8} \leq 0$. This formulation does not have an impact on the search behavior of the considered algorithms because they work with the equality constraint. The required conversion happens in a software layer between COCO and the ES implementation.

Since the objective function and the constraint function expect one input vector, the Thomson problem and the polygon problem use one vector that contains all the points. For the Thomson problem with M electrons, the implementation referenced above uses the parameterization $\mathbf{x} = (x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_M, y_M, z_M)^T$ and the evolution takes place in $\mathbb{R}^{3M}$. The objective function and the k constraints are implemented based on Equation (34). The implementation of the polygon problem with $M+1$ nodes makes use of the parameterization $\mathbf{x} = (x_1, x_2, \ldots, x_M, y_1, y_2, \ldots, y_M)^T$ and the optimization happens in $\mathbb{R}^{2M}$. The objective function and the constraint are implemented based on Equation (36).

In what follows, the performance of the various algorithm variants is visualized using bootstrapped Empirical Cumulative Distribution Functions (ECDF)[7]. Such figures show the percentages of function target values reached for a given budget of function and constraint evaluations divided by the parameter space dimensionality. On the x-axis, the sum of objective function and constraint evaluations normalized by the dimension is shown on a logarithmic scale. On the y-axis, the percentage of targets reached for the given sum of objective function and constraint evaluations is indicated. Each of the targets is defined as a particular distance from the optimum. The standard COCO targets in the feasible region are used: $f_{\text{target}} = f_{\text{opt}} + 10^k$ for 51 different values of k between -8 and 2. COCO does not define any targets in the infeasible region. The crosses indicate the medians of the sum of objective function and constraint evaluations of those instances that were not able to reach the most difficult target. The top-left corner of every plot provides details about the experiments (suite, function number, targets, and number of instances). The legend is provided at the right end, where every entry is connected to the corresponding line in the graph.

For the experiments, the default parameters stated in [7] have been used (N is the problem parameterization's dimension, i.e., $N = 3M$ for the Thomson problem and $N = 2M$ for the polygon problem): $\lambda = 4 + \lfloor 3 \ln N \rfloor$, $\mu = \lfloor \frac{\lambda}{2} \rfloor$,

[7]Results in tabular form are provided in Appendix D.

$w_m = \frac{\ln\left(\frac{\lambda+1}{2}\right) - \ln m}{\sum_{k=1}^{\mu}\left(\ln\left(\frac{\lambda+1}{2}\right) - \ln k\right)}$ for $1 \leq m \leq \mu$, $\mu_{\text{eff}} = \frac{1}{\sum_{m=1}^{\mu} w_m^2}$, $c_s = \frac{\mu_{\text{eff}}+2}{\mu_{\text{eff}}+N+5}$, $c_1 = \frac{2}{(N+1.3)^2+\mu_{\text{eff}}}$, $c_w = \min\left(1 - c_1, \frac{2(\mu_{\text{eff}}+1/\mu_{\text{eff}}-2)}{(N+2)^2+\mu_{\text{eff}}}\right)$. The budget of the sum of function and constraint evaluations has been set to 10^5N. The iterative repair accuracy has been set to $\epsilon = 10^{-9}$ and the maximum number of iterations in the repair step has been set to $T = 10$. For the polygon problem, $L = 10$ has been used in the experiments.

4.3.1. Benchmarking the Different MA-ES Variants

In the plots that follow, the abbreviations as outlined in the list below are used.

- **elli**: denotes the $(\mu/\mu_w, \lambda)$-MA-ES for optimization on ellipsoidal manifolds (Algorithm 1).
- **repfeval**: is the $(\mu/\mu_w, \lambda)$-MA-ES for optimization under general non-linear equality constraints (Algorithm 2) with the repair only performed for the objective function evaluation. The evolution itself happens in an unconstrained manner. That is, *shouldDoBackCalculation* is set to false and Line 25 is omitted.
- **backcalc1**: indicates Algorithm 2 with *shouldDoBackCalculation* set to true and *shouldDoIterativeUpdateOfMInverse* set to false.
- **backcalc2**: denotes Algorithm 2, where *shouldDoBackCalculation* and *shouldDoIterativeUpdateOfMInverse* are set to true.
- **nobackcalc**: indicates Algorithm 2 with *shouldDoBackCalculation* set to false.

Figures 2 and 3 show the ECDF plots of the different algorithm variants in comparison on the Thomson problem. Since exact optima are not known for all the different numbers of electrons considered, the values of f_{opt} used in the experiments are summarized in the appendix (see Table C.1 in Appendix C). For the targets, the corresponding relative error to the optimal value is used.

The largest performance difference is observed between the ES variant that evolves on the ellipsoidal manifold and the ES variants with the repair approach. The ellipsoidal variant reaches between about 60% and 80% of the targets with fewer evaluations than the other methods. The reason is that the iterative repair method requires constraint evaluations whereas the ellipsoidal variant assumes a known elliptic constraint and evolves itself on it. Additionally, one observes that the two update variants of $\mathbf{M}_{\text{inv}}$ exhibit similar behaviors. This suggests that the iterative $\mathbf{M}_{\text{inv}}$ update can be used instead of the full inverse computation.

Analogously, Figures 4 and 5 show the ECDF plots of the different algorithm variants in comparison on the polygon problem. Note that the way the polygon problem is stated in (36), 0 is the optimal value to be reached. Hence, for the polygon problem, the absolute error to 0 with the corresponding target's tolerance is used for each target considered.

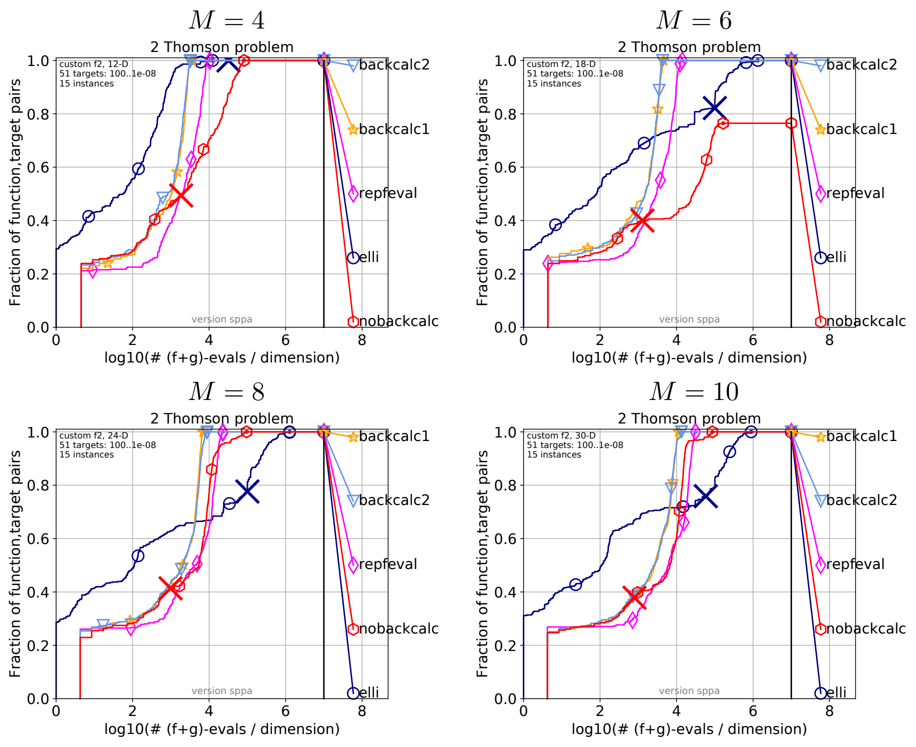

Figure 2: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of the different approaches on the Thomson problem. Notice that due to the problem's parameterization, the optimization is performed in $N = 3M$ dimensions. (Part 1/2)

The figures show that the repair without back-calculation performs worse than the variants with back-calculation. Interestingly, the difference in performance shows up already for $M = 5$, i.e., for a polygon with 6 nodes. Further investigations concerning the behavior for this case showed that a steady decrease toward the optimizer is only achieved with back-calculation. The dynamics of a single run with and without back-calculation are shown side by side in Figure 6. Furthermore, one observes again that the two update variants of $\mathbf{M}_{\text{inv}}$ exhibit similar behaviors, which indicates that the iterative $\mathbf{M}_{\text{inv}}$ update can be used.

4.3.2. Comparison With Other Approaches

The MA-ES variant with back-calculation shows the best performance in Figures 2 to 5. Since the variants with the full inverse computation and the iterative computation of $\mathbf{M}_{\text{inv}}$ perform similarly, the back-calculation variant with iterative $\mathbf{M}_{\text{inv}}$ update has been used in further experiments. It has been compared to the other evolutionary optimization methods that are described in the following paragraphs.

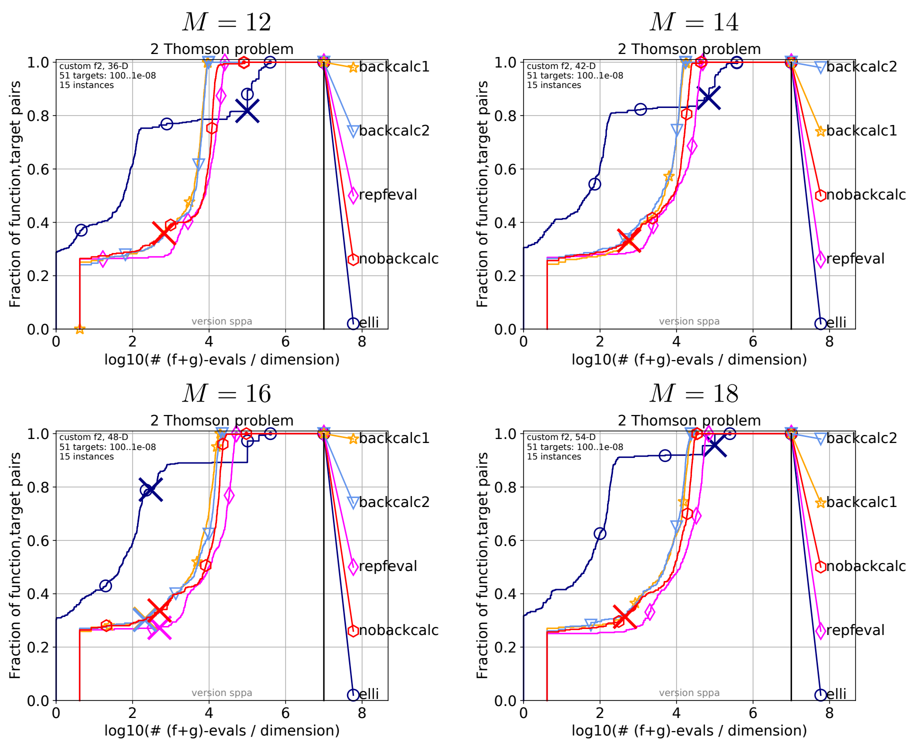

Figure 3: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of the different approaches on the Thomson problem. (Part 2/2)

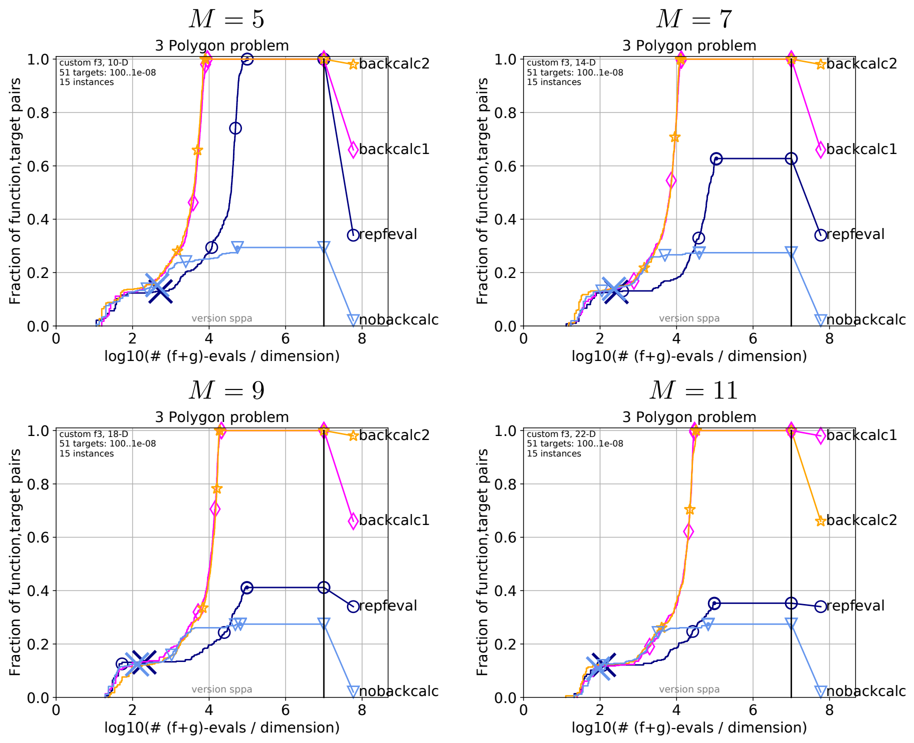

Figure 4: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of the different approaches on the polygon problem. Notice that due to the problem's parameterization, the optimization is performed in $N = 2M$ dimensions. (Part 1/2)

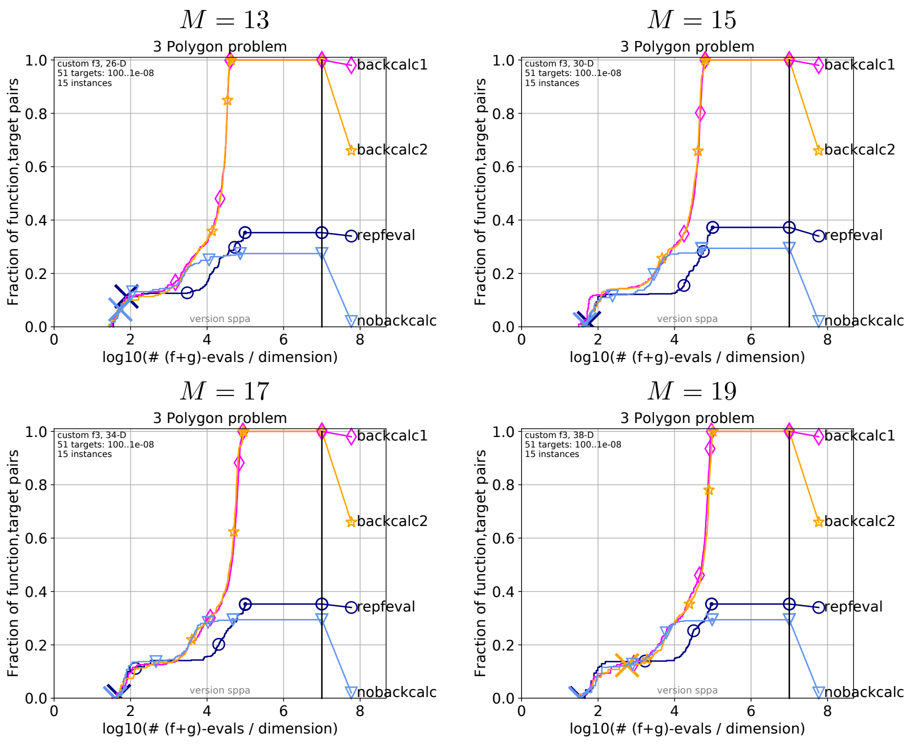

Figure 5: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison of the different approaches on the polygon problem. (Part 2/2)

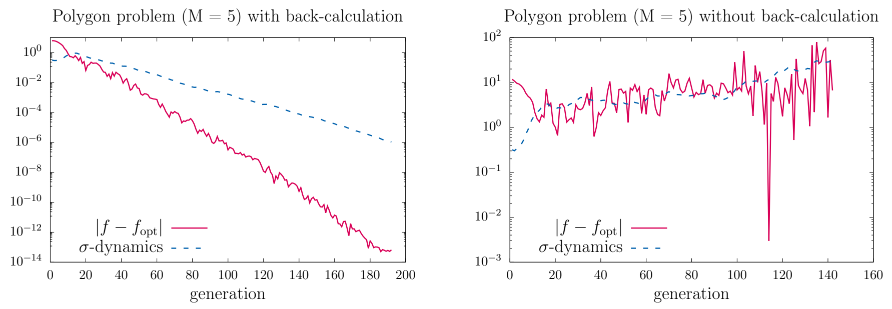

Figure 6: Dynamics of a single run of the variants with back-calculation (left) and without back-calculation (right) on the polygon problem for $M = 5$.

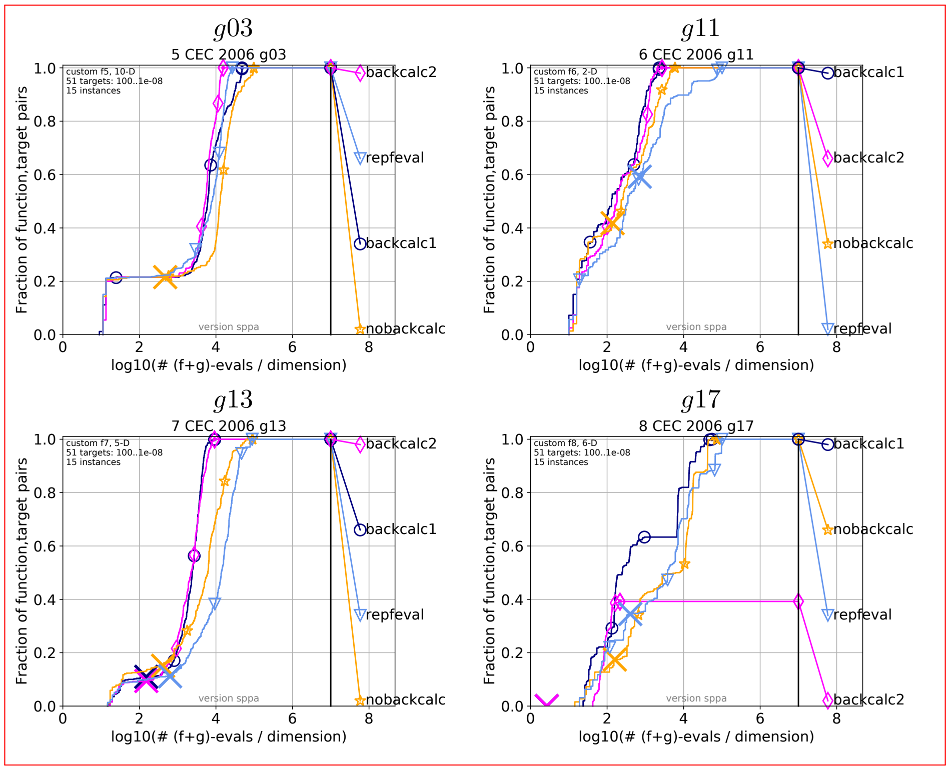

Figure 7: Bootstrapped empirical cumulative distribution function of the number of objec-tive function and constraint evaluations divided by dimension: comparison of the different approaches on the CEC 2006 problems with only nonlinear constraints.

ConSaDE (denoted `conSaDE` *in the plots).* The Self-adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization (ConSaDE) [3] is an extension of the Self-adaptive Differential Evolution (SaDE) algorithm with constraint handling. The two main aspects of the SaDE algorithm are the use of multiple mutation operators and self-adaptive parameter adaptation. Furthermore, a local search is performed every 500 generations with the idea of speeding up the convergence. The variant ConSaDE incorporates constraint handling into SaDE by using lexicographic ordering in the selection step.

ECHT-DE (denoted `ECHT-DE o` *in the plots).* Differential Evolution with Ensemble of Constraint Handling Techniques (ECHT-DE) as proposed in [4] is a combination of various constraint handling methods. The constraint handling methods used are superiority of feasibility, self-adaptive penalty, ϵ-level constraint handling, and stochastic ranking. Each of them has its own population with a set of parameters and each generates offspring. For the computation of the next generation's population all the different current populations are considered.

ϵDEag (denoted `epsDEga o` *in the plots).* The method Constrained Optimization by the ϵ Constrained Differential Evolution with an Archive and Gradient-Based Mutation (ϵDEag) [5] uses an archive to maintain diversity, incorporates the ϵ-level constraint handling approach and the gradient-based mutation method for dealing with constraints.

Active-Set-ES (denoted `active se` *in the plots).* The Active-Set-ES [17] is a $(1+1)$-ES with an evolution of the active set of constraints. In each generation, a feasible offspring is created. This is then projected either onto a reduced search space or onto the whole feasible region. That choice is performed uniformly at random with a fixed probability. In the reduced search space, all the inequality constraints that are active at the parent are turned into equality ones. For the mutation strength adaptation, the 1/5th rule is used.

ϵMAg-ES (denoted `epsMAg-ES` *in the plots).* The ϵMAg-ES [8] incorporates three constraint handling techniques into the MA-ES. It makes use of a reflection approach for dealing with the box-constraints and the ϵ-level constraint handling. Further, a repair method based on estimated gradients is used.

LSHADE44 (denoted `LSHADE44` *in the plots).* The L-SHADE with Competing Strategies Applied to Constrained Optimization (LSHADE44) has been proposed in [6]. It is the winner of the CEC 2017 competition on constrained optimization and improves the Success History based DE with linear popula-tion size reduction (L-SHADE) [25]. Four different combinations of mutation and crossover operators are used that compete with each other to create a trial point. The lexicographic ordering approach is used for the constraint handling.

iUDE (denoted `iUDE` *in the plots).* The iUDE is an improved version of the Unified Differential Evolution (UDE) [26] algorithm. The iUDE method was the winner of the CEC 2018 competition on constrained optimization[8]. The algorithm uses three trial vector generation approaches. Additionally, a dual population approach with strategy adaptation is incorporated. Lexicographic and ϵ-level constraint handling are used for dealing with the restrictions.

fmincon (denoted `fmincon` *in the plots).* Matlab's fmincon has been used with the option `interior-point` to have a further interior-point method besides the Active-Set-ES and the proposed algorithms.

ECDF plots. The corresponding comparison ECDF plots are shown in Figures 8 and 9 for the Thomson problem, in Figures 10 and 11 for the polygon problem, and in Figure 12 for the CEC 2006 problems with only nonlinear equality constraints.

They show that the proposed method is competitive. On the Thomson problem, fmincon (interior-point) exhibits the best performance and reaches the most difficult target for all dimensions shown. The proposed approach as well as the Active-Set-ES show a similar performance and reach the most difficult target (note that the Active-Set-ES requires more evaluations). For smaller dimensions, the ϵMAg-ES reaches the most difficult target, however, it requires more evaluations. It is not able to reach the most difficult target for the higher dimensions. Considering the DE variants, the conSaDE is competitive[9] up to $M = 10$. For the higher dimensions considered, the performance of all the DE variants is inferior to the ESs considered.

On the polygon problem, for the smaller dimensions the situation is similar to the Thomson problem. However, for the larger dimensions the proposed approach shows the best performance. The ϵMAg-ES reaches about 60% of the targets. All the other approaches reach less than 20% of the targets for the larger dimensions.

Considering the CEC 2006 problems with only nonlinear equality constraints, the results show that the proposed approach and the Active-Set-ES achieve best performance over all problems. In particular for g17, the others perform worse. On g03 and g13, additionally the conSaDE and the ϵMAg-ES reach the most difficult target. On g11, all methods except the iUDE reach the most difficult target.

5. Conclusion

Two different ESs have been presented for dealing with nonlinear equality constraints. The first algorithm is for optimizing a black-box function and as-

[8]A technical report and the source code (Matlab) of iUDE have been made available as part of the CEC 2018 competition materials.

[9]It is worth noting that conSaDE uses Matlab's `fmincon` at a pre-defined interval of passed generations to speed up the convergence.

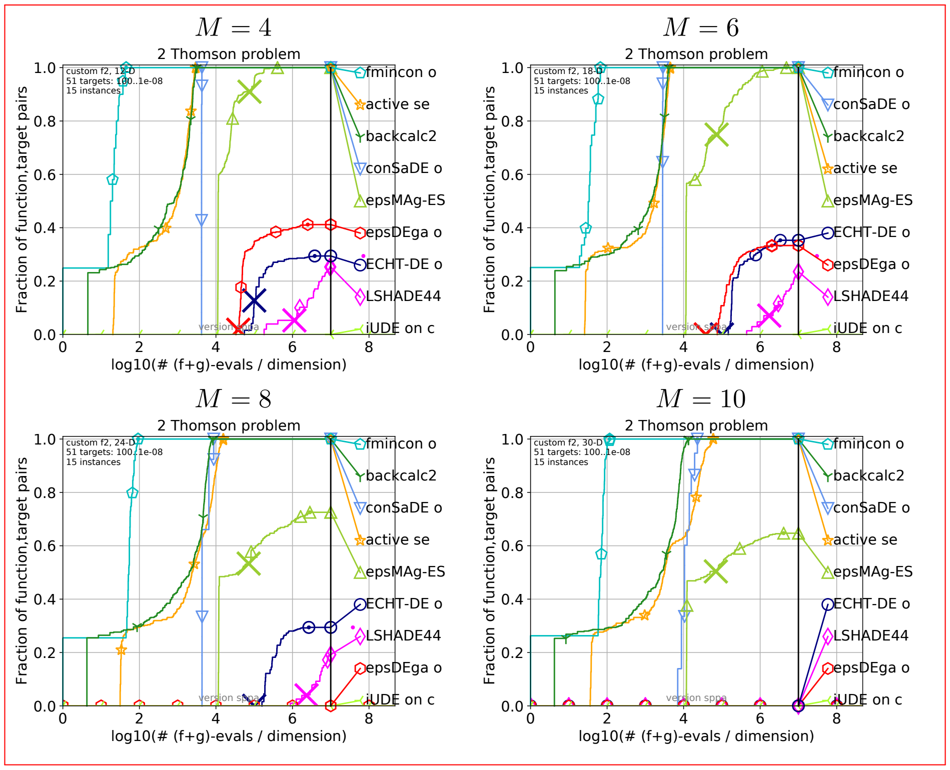

Figure 8: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison with other methods on the Thomson problem. Notice that due to the problem's parameterization, the optimization is performed in $N = 3M$ dimensions. (Part 1/2)

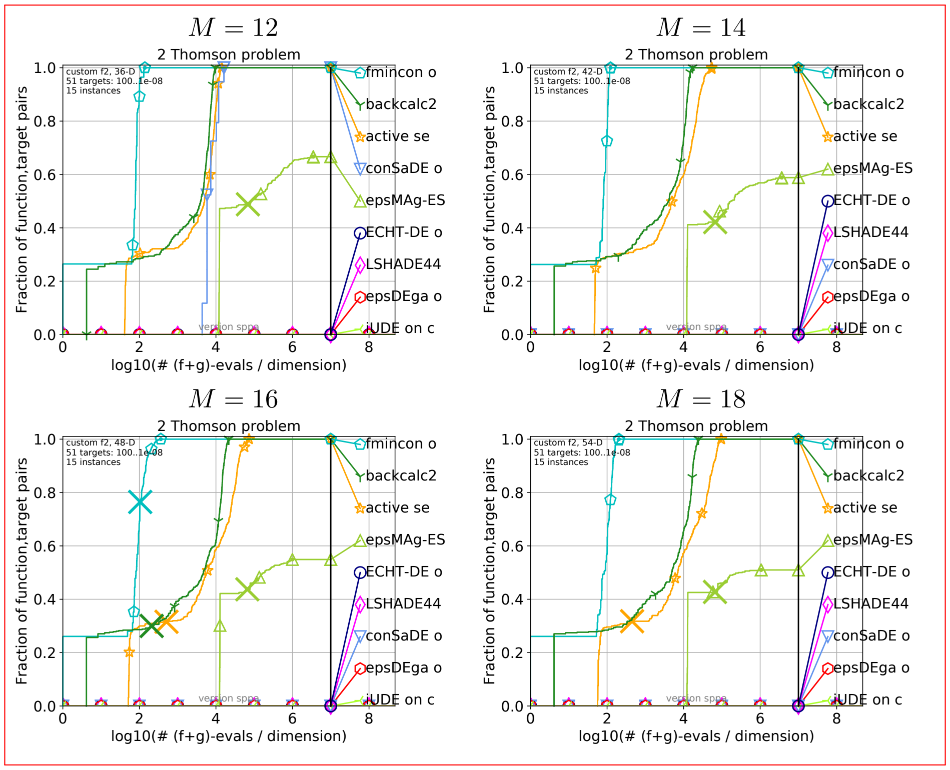

Figure 9: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison with other methods on the Thomson problem. (Part 2/2)

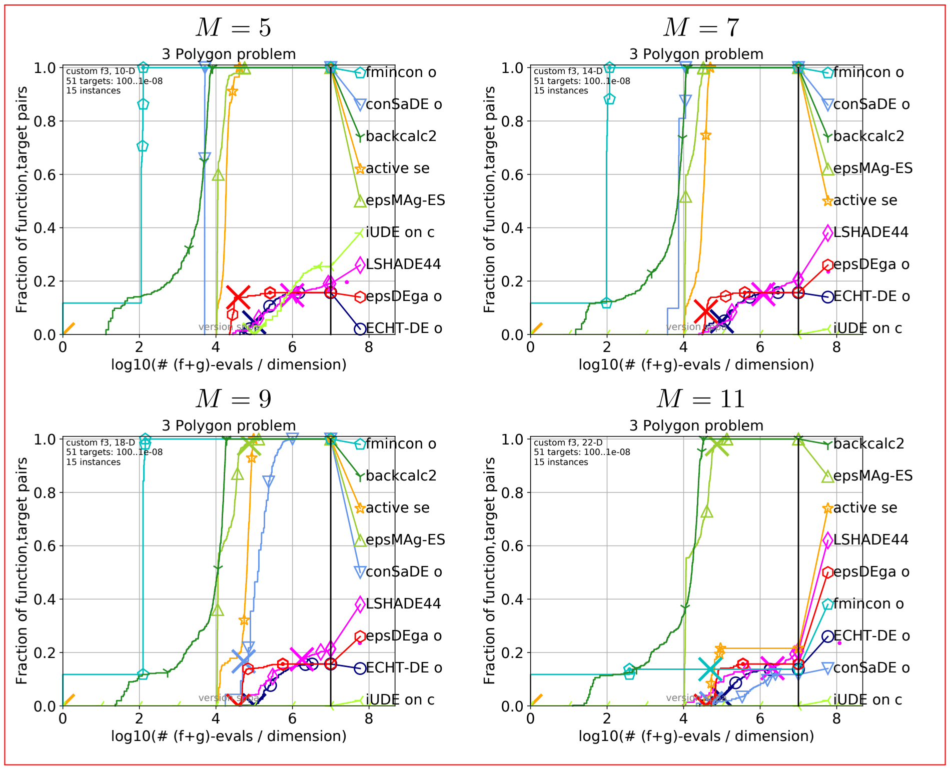

Figure 10: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison with other methods on the polygon problem. Notice that due to the problem's parameterization, the optimization is performed in $N = 2M$ dimensions. (Part 1/2)

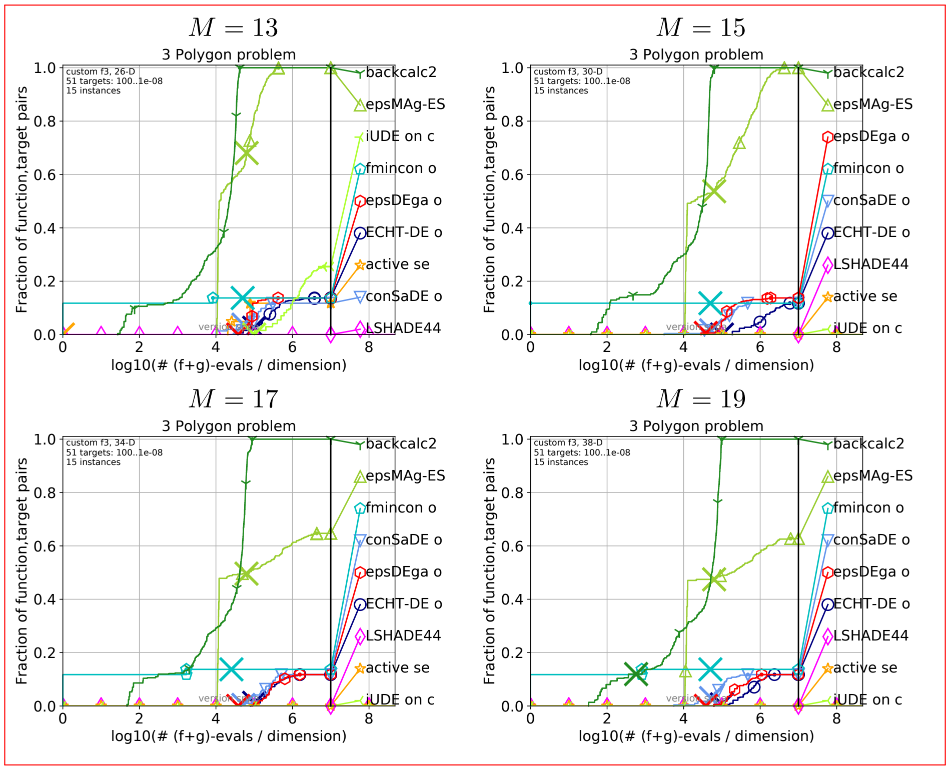

Figure 11: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison with other methods on the polygon problem. (Part 2/2)

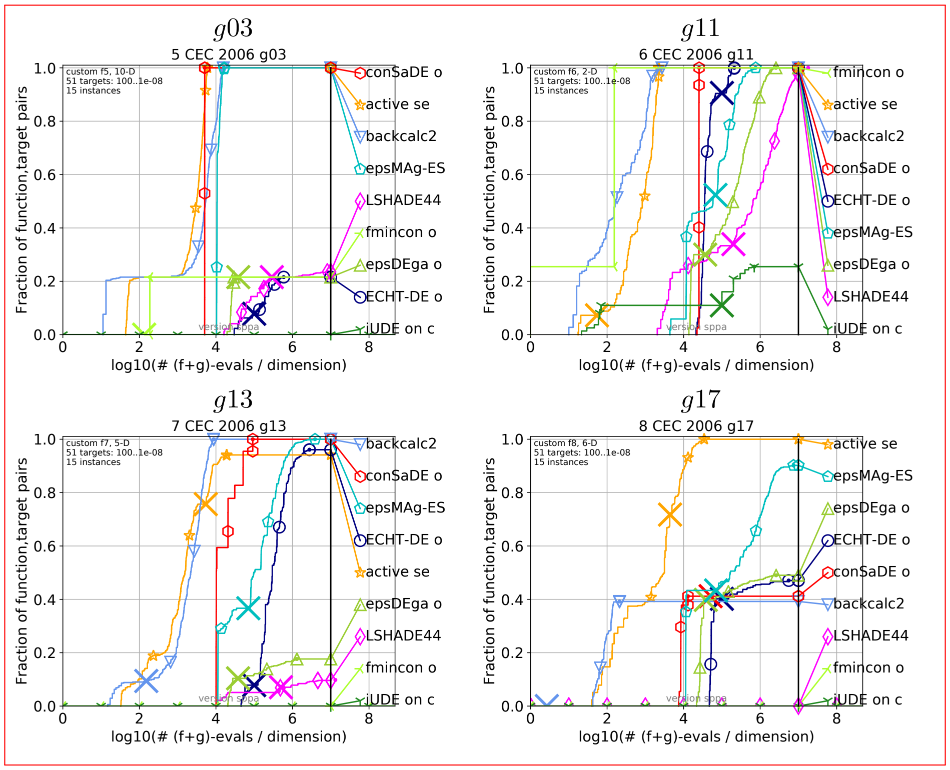

Figure 12: Bootstrapped empirical cumulative distribution function of the number of objective function and constraint evaluations divided by dimension: comparison with other methods on the CEC 2006 problems with only nonlinear equality constraints.

sumes a known constraint in the form of an elliptical manifold. The second algorithm supports general nonlinear (black-box) equality constraints optimizing a black-box objective function. Both algorithms have been evaluated on Thomson's problem and a problem of maximizing the area of a polygon given a fixed circumference.

Different variations of the algorithms have been compared. As a remarkable result, it has been experimentally shown that the back-calculation is crucial for a working ES for the consideration in this work. In particular, the experiments revealed that for the second algorithm the part that is enabled if *shouldDoBackCalculation* is set to true yields better results for the polygon problem. As a reason for it, the dynamics of a single run have been shown to yield a desired behavior only for the former case. In addition, the iterative update of the covariance factor indicated by *shouldDoIterativeUpdateOfMInverse* has not shown qualitative differences. Hence, one can conclude from the experiments that the cheaper and more robust iterative update can be preferred to the full inverse computation.

Comparisons of the variant for the case that *shouldDoBackCalculation* and *shouldDoIterativeUpdateOfMInverse* are set to true with other evolutionary optimization methods have also been performed. They show that the proposed methods are competitive on the problems considered.

Gaining even further insight into the inner workings of the algorithms is a topic for future work. If possible, theoretical investigations of parts of the algorithm can result in a deeper knowledge. Another direction for future research can be the investigation of parabolic and hyperbolic constraints. It would be of interest to study whether a closed form evolution on parabolic and hyperbolic manifolds is possible with a similar approach as for the ellipsoidal constraint presented in this work.

Appendix

A. Offspring Creation and Parental Individual Update on the Elliptical Manifold

Lines 11 and 15 of Algorithm 1 are designed such that the created offspring and the parental individual for the next generation, respectively, are on the manifold, i.e., it is ensured by design that they satisfy the constraint $\mathbf{x}^T\mathbf{S}\mathbf{x} = \varkappa$. A straightforward calculation (remembering $\mathbf{S} = \mathbf{A}^T\mathbf{A}$) shows that such an algorithm design yields the desired result:

$$\tilde{\mathbf{x}}_l^T \mathbf{S} \tilde{\mathbf{x}}_l = \varkappa \frac{\left(\mathbf{x} + \sigma \mathbf{A}^{-1} \tilde{\mathbf{d}}_l\right)^T \mathbf{A}^T \mathbf{A} \left(\mathbf{x} + \sigma \mathbf{A}^{-1} \tilde{\mathbf{d}}_l\right)}{\left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)^T \left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)} \tag{A.1}$$

$$= \varkappa \frac{\left(\mathbf{A}\left(\mathbf{x} + \sigma \mathbf{A}^{-1} \tilde{\mathbf{d}}_l\right)\right)^T \mathbf{A} \left(\mathbf{x} + \sigma \mathbf{A}^{-1} \tilde{\mathbf{d}}_l\right)}{\left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)^T \left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)} \tag{A.2}$$

$$= \varkappa \frac{\left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)^T \left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)}{\left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)^T \left(\mathbf{A}\mathbf{x} + \sigma \tilde{\mathbf{d}}_l\right)} = \varkappa. \tag{A.3}$$

This result is for the offspring creation. The same result can be shown analogously for the parental individual update.

B. Derivation of the Maximal Area $A_{\max}$

In the limit case of infinitely many points with infinitesimally small segment lengths, the goal of maximal area given a fixed circumference yields a circle. Therefore, $A_{\max}$ can be calculated as depicted in Figure B.13 by considering equal angles β for all the $P = M + 1$ segments of equal length d. The radius of the circle is denoted as R. The visualization is for the case of a polygon with $P = 4$ segments. For the general case,

$$\beta = \frac{2\pi}{P}. \tag{B.1}$$

In general, the dashed line intersects the segment with 90°. Hence, one can write

$$\sin\frac{\beta}{2} = \frac{d/2}{R}, \tag{B.2}$$

which implies

$$d = 2R\sin\frac{\beta}{2}. \tag{B.3}$$

Since all the segments have the same length, the circumference L of the polygon can be calculated as

$$L = Pd. \tag{B.4}$$

Insertion of Equation (B.3) into Equation (B.4) yields

$$L = 2PR\sin\frac{\beta}{2}, \tag{B.5}$$

which implies

$$R = \frac{L}{2P\sin\frac{\beta}{2}}. \tag{B.6}$$

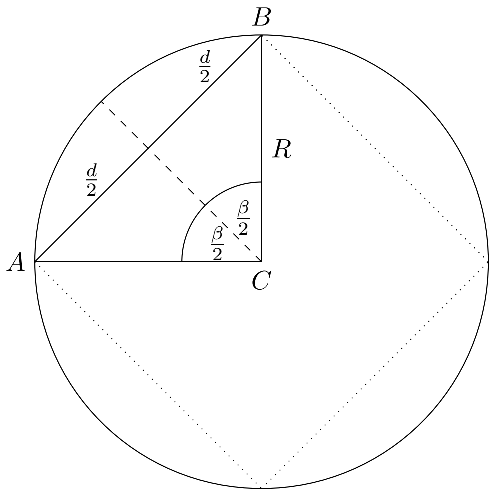

Figure B.13: Visualization for the computation of $A_{\max}$. One particular segment of a polygon with 4 segments is considered. Its length is d. It is shown as a solid line, whereas the other 3 segments are dotted. A triangle is shown with the corners C, A, and B, where C denotes the center of the depicted circle with radius R and the two corners A and B are the intersection points of the shown segment with the circle. The angle $\angle CAB$ is denoted as β. Note that in the case considered $\beta = 90°$.

Inserting Equation (B.1) into Equation (B.6) results in

$$R = \frac{L}{2P \sin \frac{\pi}{P}}. \tag{B.7}$$

The area of each segment can be calculated leading to

$$A_{\text{seg}} = \frac{d}{2} R \cos \frac{\beta}{2} \tag{B.8}$$

(refer to the triangle CAB in Figure B.13). Since there are P segments, one gets

$$A_{\max} = P A_{\text{seg}} = P \frac{d}{2} R \cos \frac{\beta}{2}. \tag{B.9}$$

Usage of Equation (B.4) and Equation (B.7) results in

$$A_{\max} = \frac{L}{2} \frac{L}{2P} \frac{\cos \frac{\beta}{2}}{\sin \frac{\beta}{2}} = \frac{L^2}{4P} \frac{1}{\tan \frac{\beta}{2}} \tag{B.10}$$

for Equation (B.9). Taking into account Equation (B.1) and $P = M + 1$, one gets

$$A_{\max} = \frac{L^2}{4P} \frac{1}{\tan \frac{\pi}{P}} = \frac{L^2}{4 \tan \frac{\pi}{M+1}} \frac{1}{M+1}. \tag{B.11}$$

M	f_{opt}
2	0.500000000
3	1.732050808
4	3.674234614
5	6.474691495
6	9.985281374
7	14.452977414
8	19.675287861
9	25.759986531
10	32.716949460
11	40.596450510
12	49.165253058
13	58.853230612
14	69.306363297
15	80.670244114
16	92.911655302
17	106.050404829
18	120.084467447

Table C.1: The optimal values used for benchmarking the Thomson problem.

C. Considered Best Known Values for the Thomson Problem

The optimal values[10] considered for benchmarking the Thomson problem are summarized in Table C.1.

D. Experimental Results in Tabular Form

Using the COCO post-processing tool, tabular results that accompany the comparison plots with other approaches shown in Section 4.3 have been generated. They are presented in the following tables: Tables D.2 to D.9 show tabular results for the Thomson problem. Tables D.10 to D.17 show tabular results for the polygon problem. Tables D.18 to D.21 show tabular results for the CEC 2006 problems g03, g11, g13, and g17.

Acknowledgments

This work was supported by the Austrian Science Fund FWF under grant P29651-N32.

[10]They are listed in `https://en.wikipedia.org/wiki/Thomson_problem#Configurations_of_smallest_known_energy` (accessed: June 14, 2019) and present best known values.

M=4

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	1.4e6(6e5)	1.4e6(6e5)	∞	∞	∞	∞	∞ *1e6*	0/15
LSHADE4	5.7e7(5e7)	5.7e7(7e7)	∞	∞	∞	∞	∞ *1e7*	0/15
active	262(14)	275(44)	**981**(926)	6824(1840)	1.2e4(2081)	**2.0e4**(1935)	**2.8e4**(3352)	15/15
conSaDE	5.1e4(122)	5.1e4(532)	5.1e4(328)	5.1e4(322)	5.1e4(338)	5.1e4(556)	5.1e4(126)	15/15
epsDEga	5.3e5(9e4)	5.9e5(9e4)	7.2e5(1e5)	7.2e6(1e7)	∞	∞	∞ *5e5*	0/15
epsMAg-	1.4e5(3238)	1.4e5(3404)	2.0e5(2e5)	2.0e5(2e5)	2.0e5(2e5)	3.5e5(6e5)	6.6e5(5e5)	10/15
fmincon	**1**(0)$^{\star 5}$	**79**(128)	**269**(128)$^{\star 2}$	**283**(148)$^{\star 5}$	**283**(107)$^{\star 5}$	**283**(78)$^{\star 5}$	**286**(114)$^{\star 5}$	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *1e6*	0/15
backcal	**54**(0)	**76**(54)	1494(1235)	**4236**(2592)	**1.0e4**(4890)	2.1e4(3740)	3.0e4(3422)	15/15

Table D.2: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 4$. For each function, the aRT and, in braces as dispersion measure, the half difference between 10 and 90%-tile of (bootstrapped) runtimes is shown for the different target Δf-values as shown in the top row. #succ is the number of trials that reached the last target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function and constraint evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1. Best results are printed in bold.

M=6

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	3.6e6(1e6)	4.1e6(8e5)	2.8e7(4e7)	∞	∞	∞	∞ *2e6*	0/15
LSHADE4	1.0e8(9e7)	1.0e8(8e7)	∞	∞	∞	∞	∞ *3e7*	0/15
active	485(21)	497(34)	**1764**(2226)	2.1e4(2762)	3.0e4(3014)	4.9e4(4580)	6.7e4(3501)	15/15
conSaDE	5.1e4(500)	5.1e4(320)	5.1e4(91)	5.1e4(532)	5.1e4(137)	5.1e4(112)	**5.1e4**(132)	15/15
epsDEga	2.3e6(8e5)	2.3e6(2e6)	6.0e6(6e6)	∞	∞	∞	∞ *7e5*	0/15
epsMAg-	2.1e5(1655)	2.1e5(2720)	3.0e5(3e5)	3.4e5(1e5)	4.3e5(4e5)	1.3e6(8e5)	4.6e6(6e6)	1/15
fmincon	**1**(0)$^{\star 5}$	**62**(229)$^{\star 4}$	**621**(239)$^{\star 2}$	**677**(334)$^{\star 5}$	**677**(252)$^{\star 5}$	**677**(380)$^{\star 5}$	**693**(265)$^{\star 5}$	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
backcal	**78**(0)	**94**(20)	1802(1920)	**1.3e4**(5658)	**2.7e4**(6556)	**4.7e4**(5538)	6.3e4(8448)	15/15

Table D.3: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 6$.

M=8

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	7.2e6(2e6)	8.1e6(3e6)	∞	∞	∞	∞	∞ *2e6*	0/15
LSHADE4	1.9e8(3e8)	1.9e8(2e8)	∞	∞	∞	∞	∞ *6e7*	0/15
active	780(30)	797(52)	**3580**(1570)	3.8e4(8124)	5.9e4(7624)	1.6e5(6e4)	2.3e5(8e4)	15/15
conSaDE	1.3e5(3e4)	1.3e5(3e4)	1.3e5(3e4)	1.3e5(3e4)	1.3e5(3e4)	1.3e5(3e4)	**1.3e5**(3e4)	15/15
epsDEga	∞	∞	∞	∞	∞	∞	∞ *9e5*	0/15
epsMAg-	2.8e5(6920)	2.8e5(3698)	4.0e5(4e5)	5.4e5(7668)	5.9e5(4e5)	2.6e7(3e7)	∞ *2e6*	0/15
fmincon	**1**(0)$^{\star 5}$	**86**(318)$^{\star 4}$	**1277**(187)	**1368**(355)$^{\star 5}$	**1368**(328)$^{\star 5}$	**1416**(401)$^{\star 5}$	**1507**(389)$^{\star 5}$	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
backcal	**102**(0)	**109**(0)	3784(4042)	**1.6e4**(5815)	**5.1e4**(1e4)	**1.1e5**(3e4)	1.5e5(3e4)	15/15

Table D.4: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 8$.

M=10

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *3240*	0/15
active	1135(32)	1135(27)	**5382**(5631)	5.8e4(1e4)	9.4e4(2e4)	5.3e5(2e5)	8.7e5(4e5)	15/15
conSaDE	3.6e5(2e5)	3.6e5(1e5)	3.6e5(9e4)	3.6e5(2e5)	3.6e5(2e5)	3.6e5(1e5)	3.6e5(9e4)	15/15
epsDEga	∞	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsMAg-	3.6e5(5898)	3.6e5(5931)	4.7e5(4e5)	7.4e5(2e6)	2.1e6(1e6)	∞	∞ *2e6*	0/15
fmincon	**1**(0)^{*5}	**138**(0)	**1765**(671)^{*2}	**2064**(421)^{*5}	**2064**(237)^{*5}	**2278**(638)^{*5}	**2527**(370)^{*5}	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
backcal	**126**(0)	**126**(0)	7073(8032)	**2.8e4**(1e4)	**8.0e4**(2e4)	**1.8e5**(7e4)	**2.6e5**(6e4)	15/15

Table D.5: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 10$.

M=12

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	∞	∞	∞	∞	∞	∞	∞ *4e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *3888*	0/15
active	1543(60)	1543(16)	**6366**(7313)	9.3e4(1e4)	2.0e5(5e4)	2.9e5(5e4)	3.8e5(7e4)	15/15
conSaDE	2.8e5(1e5)	2.8e5(7e4)	2.8e5(2e5)	2.8e5(1e5)	2.8e5(3e4)	2.8e5(9e4)	**2.8e5**(2e5)	15/15
epsDEga	∞	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsMAg-	4.4e5(8911)	4.4e5(7726)	4.4e5(6293)	4.4e5(6883)	5.9e6(9e6)	∞	∞ *2e6*	0/15
fmincon	**1**(0)^{*5}	**210**(1566)	**2870**(402)	**2870**(358)^{*5}	**3023**(942)^{*5}	**3061**(694)^{*5}	**3185**(472)^{*5}	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *4e6*	0/15
backcal	**150**(0)	**150**(0)	1.0e4(7256)	**4.8e4**(2e4)	**1.5e5**(3e4)	**2.2e5**(3e4)	2.8e5(1e4)	15/15

Table D.6: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 12$.

M=14

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	∞	∞	∞	∞	∞	∞	∞ *4e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *4536*	0/15
active	2025(52)	2025(55)	1.1e4(6485)	1.2e5(7368)	2.3e5(1e4)	7.2e5(3e5)	1.1e6(3e5)	15/15
conSaDE	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsDEga	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsMAg-	5.2e5(1e4)	5.2e5(1e4)	7.1e5(1e4)	2.4e6(3e6)	1.2e7(1e7)	∞	∞ *3e6*	0/15
fmincon	**1**(0)^{*5}	**146**(1088)^{*4}	**3596**(1233)^{*2}	**3596**(606)^{*5}	**3596**(1082)^{*5}	**3761**(805)^{*5}	**4052**(670)^{*5}	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *4e6*	0/15
backcal	**174**(0)	**174**(0)	**1.1e4**(7526)	**4.7e4**(2e4)	**1.7e5**(4e4)	**4.0e5**(1e5)	**5.2e5**(1e5)	15/15

Table D.7: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 14$.

M=16

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	∞	∞	∞	∞	∞	∞	∞ *5e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *5184*	0/15
active	2587(88)	2587(91)	**1.1e4**(3352)	1.6e5(1e4)	2.8e5(7e4)	1.2e6(7e5)	1.8e6(8e5)	12/15
conSaDE	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
epsDEga	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsMAg-	6.1e5(1e4)	6.1e5(1e4)	6.1e5(5911)	1.4e6(8e5)	1.3e7(1e7)	∞	∞ *3e6*	0/15
fmincon	**1**(0)^{*5}	**1**(0)^{*5}	**3495**(890)^{*3}	**3766**(744)^{*5}	**3766**(811)^{*5}	**5760**(1180)^{*4}	**6473**(2798)	13/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *5e6*	0/15
backcal	**198**(0)	**211**(99)	1.2e4(1e4)	**6.2e4**(3e4)	**2.3e5**(4e4)	**6.0e5**(1e5)	**7.7e5**(9e4)	9/15

Table D.8: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 16$.

M=18

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f2								
ECHT-DE	∞	∞	∞	∞	∞	∞	∞ *5e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *5832*	0/15
active	3263(239)	3263(314)	1.5e4(9600)	2.1e5(2e4)	4.1e5(1e5)	1.7e6(9e5)	3.1e6(1e6)	14/15
conSaDE	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
epsDEga	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsMAg-	6.8e5(2e4)	6.8e5(2e4)	9.3e5(2e4)	1.8e6(4e6)	2.9e7(3e7)	∞	∞ *4e6*	0/15
fmincon	**1**(0)$^{\star 5}$	**1**(0)$^{\star 5}$	**4188**(2788)$^{\star 2}$	**4514**(1169)$^{\star 5}$	**4514**(1388)$^{\star 5}$	**5716**(1515)$^{\star 5}$	**7364**(1336)	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *5e6*	0/15
backcal	**222**(0)	**222**(0)	**1.3e4**(7465)	**7.5e4**(1e4)	**2.9e5**(4e4)	**6.8e5**(2e5)	**9.6e5**(2e5)	15/15

Table D.9: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 18$.

M=5

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	2.3e6(2e6)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
LSHADE4	2.0e6(2e6)	∞	∞	∞	∞	∞	∞ *1e7*	0/15
active	1.5e5(5e4)	1.8e5(1e5)	1.7e5(4e4)	1.8e5(4e4)	1.9e5(6e4)	2.0e5(6e4)	2.1e5(6e4)	7/15
conSaDE	5.1e4(328)	5.1e4(126)	5.1e4(332)	5.1e4(332)	5.1e4(325)	**5.1e4**(333)	**5.1e4**(523)	15/15
epsDEga	2.7e5(2e4)	∞	∞	∞	∞	∞	∞ *4e5*	0/15
epsMAg-	1.1e5(3669)	1.1e5(3981)	1.1e5(4646)	1.1e5(2798)	1.1e5(3251)	1.5e5(3e4)	2.0e5(3e4)	15/15
fmincon	**1**(0)$^{\star 5}$	**1114**(0)	**1114**(0)	**1114**(0)	**1114**(0)	**1186**(0)	**1258**(0)	15/15
iUDE on	7.5e6(7e6)	7.5e6(8e6)	∞	∞	∞	∞	∞ *1e6*	0/15
backcal	**381**(407)	**7074**(4154)	**1.8e4**(7236)	**3.0e4**(9938)	**3.8e4**(1e4)	5.3e4(4763)	6.4e4(3702)	15/15

Table D.10: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 5$. For each function, the aRT and, in braces as dispersion measure, the half difference between 10 and 90%-tile of (bootstrapped) runtimes is shown for the different target Δf-values as shown in the top row. #succ is the number of trials that reached the last target $f_{\mathrm{opt}} + 10^{-8}$. The median number of conducted function and constraint evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1. Best results are printed in bold.

M=7

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	1.9e6(2e6)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
LSHADE4	2.1e6(1e6)	2.6e8(3e8)	∞	∞	∞	∞	∞ *2e7*	0/15
active	1.7e5(2e4)	3.3e5(1e5)	4.0e5(9e4)	4.3e5(1e5)	4.6e5(8e4)	5.1e5(7e4)	5.5e5(1e5)	13/15
conSaDE	1.1e5(1e4)	1.1e5(3e4)	1.1e5(5e4)	1.1e5(4e4)	1.1e5(3e4)	**1.1e5**(1e4)	**1.1e5**(3e4)	15/15
epsDEga	5.3e5(1e5)	∞	∞	∞	∞	∞	∞ *5e5*	0/15
epsMAg-	1.5e5(3999)	1.5e5(3042)	1.5e5(4422)	1.5e5(3704)	1.5e5(4212)	2.8e5(6e4)	3.5e5(7e4)	15/15
fmincon	**1**(0)$^{\star 5}$	**1351**(0)	**1351**(0)	**1351**(0)	**1351**(0)	**1382**(0)	**1627**(0)	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *1e6*	0/15
backcal	**1417**(1961)	**1.7e4**(8005)	**4.9e4**(1e4)	**7.6e4**(1e4)	**9.6e4**(2e4)	1.3e5(1e4)	1.5e5(1e4)	15/15

Table D.11: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 7$.

M=9

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	5.1e6(9e6)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
LSHADE4	2.8e6(2e6)	1.2e8(7e7)	∞	∞	∞	∞	∞ *3e7*	0/15
active	2.3e5(1e4)	8.7e5(1e5)	9.8e5(1e5)	1.0e6(6e4)	1.1e6(1e5)	1.3e6(7e4)	1.5e6(1e5)	3/15
conSaDE	2.1e6(2e6)	2.9e6(4e6)	2.9e6(3e6)	2.9e6(2e6)	2.9e6(2e6)	2.9e6(3e6)	2.9e6(2e6)	5/15
epsDEga	9.0e5(1e5)	∞	∞	∞	∞	∞	∞ *7e5*	0/15
epsMAg-	2.0e5(8633)	2.0e5(8309)	2.0e5(7168)	2.0e5(6509)	**2.0e5**(4598)	5.4e5(1e5)	7.3e5(5e5)	14/15
fmincon	**1**(0)$^{\star 5}$	**2243**(0)$^{\star 2}$	**2243**(0)$^{\star 2}$	**2243**(0)$^{\star 2}$	**2243**(0)$^{\star 2}$	**2243**(0)$^{\star 2}$	**2354**(0)$^{\star 2}$	15/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
backcal	**2486**(1849)	**3.4e4**(7294)	**1.0e5**(2e4)	**1.6e5**(1e4)	2.0e5(2e4)	**2.6e5**(2e4)	**3.0e5**(2e4)	15/15

Table D.12: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 9$.

M=11

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	3.0e6(4e6)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
LSHADE4	2.1e6(7e5)	1.1e8(1e8)	∞	∞	∞	∞	∞ *4e7*	0/15
active	5.6e5(3e5)	1.6e6(1e5)	1.8e6(5e4)	2.0e6(9587)	∞	∞	∞ *44*	0/15
conSaDE	2.5e6(2e6)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsDEga	1.2e6(2e5)	∞	∞	∞	∞	∞	∞ *8e5*	0/15
epsMAg-	2.2e5(1e4)	**2.2e5**(9455)	**2.2e5**(9956)	**2.2e5**(9447)	**2.3e5**(5e4)	**6.9e5**(2e5)	**9.0e5**(3e5)	15/15
fmincon	**1**(0)$^{\star 5}$	∞	∞	∞	∞	∞	∞ *1e6*	0/15
iUDE on	3.0e7(7e7)	3.0e7(6e7)	∞	∞	∞	∞	∞ *2e6*	0/15
backcal	**1383**(2398)	**3.6e4**(2e4)	**1.2e5**(5e4)	**2.0e5**(4e4)	**2.6e5**(5e4)	**3.5e5**(6e4)	**4.0e5**(4e4)	15/15

Table D.13: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the Thomson problem with $M = 11$.

M=13

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	5.8e6(5e6)	∞	∞	∞	∞	∞	∞ *3e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *2808*	0/15
active	1.3e6(7e5)	∞	∞	∞	∞	∞	∞ *9*	0/15
conSaDE	2.9e6(3e6)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsDEga	2.2e6(2e5)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsMAg-	3.0e5(3e4)	**3.0e5**(2e4)	**3.0e5**(3e4)	**3.0e5**(1e4)	**3.3e5**(7e4)	**2.6e6**(2e6)	**3.5e6**(4e6)	8/15
fmincon	**1**(0)$^{\star 5}$	∞	∞	∞	∞	∞	∞ *1e6*	0/15
iUDE on	3.9e7(2e7)	3.9e7(3e7)	∞	∞	∞	∞	∞ *3e6*	0/15
backcal	**7347**(9790)	**7.3e4**(5e4)	**2.3e5**(9e4)	**4.5e5**(9e4)	**6.0e5**(8e4)	**8.0e5**(2e4)	**9.2e5**(6e4)	15/15

Table D.14: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 13$.

M=15

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	4.8e7(7e7)	∞	∞	∞	∞	∞	∞ *3e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *3240*	0/15
active	∞	∞	∞	∞	∞	∞	∞ *9*	0/15
conSaDE	4.4e6(5e6)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsDEga	4.0e6(3e6)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsMAg-	3.5e5(2e4)	**3.5e5**(2e4)	**3.5e5**(2e4)	**3.5e5**(2e4)	**2.1e6**(3e6)	**1.5e7**(1e7)	**2.9e7**(2e7)	1/15
fmincon	**1**(0)$^{\star 5}$	∞	∞	∞	∞	∞	∞ *2e6*	0/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
backcal	**7373**(1e4)	**9.1e4**(5e4)	**3.5e5**(1e5)	**7.4e5**(3e5)	**9.7e5**(2e5)	**1.3e6**(8e4)	**1.5e6**(2e5)	15/15

Table D.15: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 15$.

M=17

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	1.3e7(1e7)	∞	∞	∞	∞	∞	∞ *3e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *3672*	0/15
active	∞	∞	∞	∞	∞	∞	∞ *9*	0/15
conSaDE	7.3e6(1e7)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsDEga	1.2e7(5e6)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsMAg-	3.9e5(2e4)	**3.9e5**(2e4)	**3.9e5**(2e4)	**3.9e5**(1e4)	**3.4e6**(2e6)	∞	∞ *2e6*	0/15
fmincon	**1**(0)$^{\star 5}$	∞	∞	∞	∞	∞	∞ *9e5*	0/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
backcal	**1.0e4**(4678)	**1.4e5**(4e4)	**4.7e5**(2e5)	**1.1e6**(6e5)	**1.4e6**(1e5)	**1.9e6**(1e5)	**2.2e6**(2e5)	15/15

Table D.16: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 17$.

M=19

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f3								
ECHT-DE	3.0e7(4e7)	∞	∞	∞	∞	∞	∞ *4e6*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *4104*	0/15
active	∞	∞	∞	∞	∞	∞	∞ *9*	0/15
conSaDE	3.6e6(5e6)	∞	∞	∞	∞	∞	∞ *2e6*	0/15
epsDEga	1.2e7(1e7)	∞	∞	∞	∞	∞	∞ *1e6*	0/15
epsMAg-	4.4e5(3e4)	**4.4e5**(4e4)	**4.4e5**(4e4)	**4.4e5**(3e4)	**7.3e6**(6e6)	∞	∞ *2e6*	0/15
fmincon	**1**(0)$^{\star 5}$	∞	∞	∞	∞	∞	∞ *2e6*	0/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *4e6*	0/15
backcal	**1.4e4**(2e4)	**1.7e5**(7e4)	**6.5e5**(5e5)	**1.6e6**(7e5)	**2.2e6**(7e5)	**3.0e6**(5e5)	**3.2e6**(5e5)	12/15

Table D.17: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the polygon problem with $M = 19$.

g03

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f5								
ECHT-DE	1.7e6(2e6)	1.7e6(9e5)	∞	∞	∞	∞	∞ *1e6*	0/15
LSHADE4	8.9e5(8e5)	8.9e5(6e5)	∞	∞	∞	∞	∞ *3e6*	0/15
active	**492**(45)	**601**(91)	**1.9e4**(1684)	**2.6e4**(1874)	**3.1e4**(2200)	**4.2e4**(3373)	**5.3e4**(3567)	15/15
backcal	**127**(11)$^{\star 5}$	**211**(73)$^{\star 5}$	**5.0e4**(5e4)	5.8e4(4e4)	6.7e4(4e4)	7.9e4(3e4)	8.9e4(3e4)	15/15
conSaDE	5.1e4(59)	5.1e4(136)	5.1e4(134)	**5.1e4**(85)	**5.1e4**(44)	**5.1e4**(148)	**5.1e4**(93)	15/15
epsDEga	2.5e5(2e4)	2.5e5(2e4)	∞	∞	∞	∞	∞ *4e5*	0/15
fmincon	1.1e5(2027)	1.1e5(1663)	1.1e5(1935)	1.1e5(2108)	1.1e5(2164)	1.2e5(2e4)	1.3e5(2e4)	15/15
iUDE on	1874(0)	1874(0)	∞	∞	∞	∞	∞ *1333*	0/15
backcal	∞	∞	∞	∞	∞	∞	∞ *1e6*	0/15

Table D.18: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the CEC 2006 problem g03. For each function, the aRT and, in braces as dispersion measure, the half difference between 10 and 90%-tile of (bootstrapped) runtimes is shown for the different target Δf-values as shown in the top row. #succ is the number of trials that reached the last target $f_{\text{opt}} + 10^{-8}$. The median number of conducted function and constraint evaluations is additionally given in *italics*, if the target in the last column was never reached. Entries, succeeded by a star, are statistically significantly better (according to the rank-sum test) when compared to all other algorithms of the table, with $p = 0.05$ or $p = 10^{-k}$ when the number k following the star is larger than 1. Best results are printed in bold.

g11

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f6								
ECHT-DE	7.9e4(8e4)	7.9e4(2e5)	8.8e4(3e4)	8.8e4(1e5)	8.9e4(7e4)	1.1e5(2e4)	1.3e5(7e4)	9/15
LSHADE4	1.9e4(3808)	1.9e4(1988)	1.1e6(1e6)	2.9e6(2e6)	5.6e6(3e6)	5.6e6(6e6)	5.6e6(5e6)	1/15
active	503(438)	516(394)	884(794)	1844(1395)	2178(1431)	2623(1203)	3155(1072)	14/15
backcal	**34**(16)	**54**(78)	**131**(122)$^{\star 3}$	**229**(150)*	**561**(827)	**1579**(1250)	**2341**(1027)	15/15
conSaDE	5.0e4(220)	5.0e4(79)	5.0e4(22)	5.0e4(159)	5.0e4(152)	5.0e4(154)	5.0e4(22)	15/15
epsDEga	3.1e4(2618)	3.1e4(2224)	1.8e5(2e5)	5.5e5(1e6)	1.1e6(2e6)	1.1e6(2e6)	1.2e6(6e5)	1/15
fmincon	2.2e4(7578)	2.2e4(3847)	8.1e4(1e5)	1.5e5(2e5)	2.2e5(2e5)	3.1e5(3e5)	3.1e5(4e5)	6/15
iUDE on	**1**(0)$^{\star 5}$	**1**(0)$^{\star 5}$	**308**(0)	**308**(0)	**308**(0)	**308**(0)$^{\star 4}$	**308**(0)$^{\star 5}$	15/15
backcal	2.3e5(4e5)	2.3e5(5e5)	∞	∞	∞	∞	∞ *2e5*	0/15

Table D.19: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the CEC 2006 problem g11.

g13

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f7								
ECHT-DE	7.7e5(7e4)	1.3e6(2e6)	1.5e6(1e6)	1.9e6(2e6)	2.3e6(2e6)	2.3e6(3e6)	3.0e6(1e6)	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *2e6*	0/15
active	**2772**(711)	**9276**(2e4)	**1.1e4**(3e4)	**1.2e4**(2e4)	**1.3e4**(9227)	**1.6e4**(2e4)	**2.2e4**(2e4)	0/15
backcal	**1925**(1974)	**5050**(3754)	**8617**(5126)	**1.1e4**(5180)	**1.3e4**(9460)	**1.9e4**(1e4)	**2.2e4**(1e4)	5/15
conSaDE	5.5e4(404)	1.2e5(2e5)	1.2e5(6e4)	1.2e5(2e5)	1.2e5(1e5)	1.2e5(5e4)	1.2e5(3e4)	15/15
epsDEga	5.0e5(7e5)	∞	∞	∞	∞	∞	∞ *2e5*	0/15
fmincon	2.9e5(6e5)	5.8e5(5e5)	7.6e5(8e5)	7.6e5(2e6)	7.6e5(4e5)	1.5e6(2e6)	2.5e6(3e6)	2/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *7*	0/15
backcal	∞	∞	∞	∞	∞	∞	∞ *5e5*	0/15

Table D.20: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the CEC 2006 problem g13.

g17

Δf_{opt}	1e1	1e0	1e-1	1e-2	1e-3	1e-5	1e-7	#succ
f8								
ECHT-DE	3.2e5(4e4)	3.2e5(4e4)	3.2e5(3e4)	1.2e6(1e6)	∞	∞	∞ *6e5*	0/15
LSHADE4	∞	∞	∞	∞	∞	∞	∞ *3e6*	0/15
active	**877**(553)	**877**(912)	**877**(804)	**9606**(7330)	**3.7e4**(3e4)	**5.0e4**(5e4)	**5.0e4**(4e4)	7/15
backcal	**514**(246)	**514**(280)	**514**(204)	∞	∞	∞	∞ *16*	0/15
conSaDE	5.2e4(2611)	5.2e4(1613)	5.2e4(1261)	**9.4e4**(2e5)	∞	∞	∞ *3e5*	0/15
epsDEga	1.6e5(3e4)	1.6e5(2e4)	1.6e5(3e4)	6.3e5(5e5)	∞	∞	∞ *2e5*	0/15
fmincon	6.9e4(1484)	6.9e4(1615)	6.9e4(1200)	9.9e4(893)	**6.0e6**(7e6)	**6.0e6**(9e6)	**6.1e6**(6e6)	0/15
iUDE on	∞	∞	∞	∞	∞	∞	∞ *31*	0/15
backcal	∞	∞	∞	∞	∞	∞	∞ *6e5*	0/15

Table D.21: Average runtime (aRT) to reach given targets, measured in number of function and constraint evaluations for the CEC 2006 problem g17.

References

[1] E. Mezura-Montes, C. A. C. Coello, Constraint-handling in nature-inspired numerical optimization: Past, present and future, Swarm and Evolutionary Computation 1 (4) (2011) 173–194.

[2] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.

[3] V. L. Huang, A. K. Qin, P. N. Suganthan, Self-adaptive differential evolution algorithm for constrained real-parameter optimization, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2006, pp. 17–24.

[4] R. Mallipeddi, P. N. Suganthan, Differential evolution with ensemble of constraint handling techniques for solving CEC 2010 benchmark problems, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–8.

[5] T. Takahama, S. Sakai, Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation, in: IEEE Congress on Evolutionary Computation (CEC), IEEE, 2010, pp. 1–9.

[6] R. Poláková, L-SHADE with competing strategies applied to constrained optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1683–1689. doi:10.1109/CEC.2017.7969504.

[7] H.-G. Beyer, B. Sendhoff, Simplify your covariance matrix adaptation evolution strategy, IEEE Transactions on Evolutionary Computation 21 (5) (2017) 746–759.

[8] M. Hellwig, H.-G. Beyer, A matrix adaptation evolution strategy for constrained real-parameter optimization, in: 2018 IEEE Congress on Evolutionary Computation (CEC), IEEE, Rio de Janeiro, Brazil, Brazil, 2018, pp. 1–8. doi:10.1109/CEC.2018.8477950.
URL https://ieeexplore.ieee.org/document/8477950

[9] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.

[10] N. Hansen, S. D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evolutionary Computation 11 (1) (2003) 1–18.

[11] H.-G. Beyer, S. Finck, T. Breuer, Evolution on trees: On the design of an evolution strategy for scenario-based multi-period portfolio optimization under transaction costs, Swarm and Evolutionary Computation 17 (2014) 74–87.

[12] S. Finck, Worst case search over a set of forecasting scenarios applied to financial stress-testing, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '19, ACM, New York, NY, USA, 2019, pp. 1722–1730.

[13] P. Spettel, H.-G. Beyer, M. Hellwig, A covariance matrix self-adaptation evolution strategy for optimization under linear constraints, IEEE Transactions on Evolutionary Computation 23 (3) (2019) 514–524. `doi:10.1109/TEVC.2018.2871944`.

[14] J. J. Thomson, On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science 7 (39) (1904) 237–265. `doi:10.1080/14786440409463107`.
URL `https://doi.org/10.1080/14786440409463107`

[15] P. M. L. Tammes, On the origin of number and arrangement of the places of exit on the surface of pollen-grains, Ph.D. thesis, University of Groningen, relation: https://www.rug.nl/ Rights: De Bussy (1930).

[16] D. V. Arnold, On the use of evolution strategies for optimization on spherical manifolds, in: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (Eds.), Parallel Problem Solving from Nature – PPSN XIII, Springer International Publishing, Cham, 2014, pp. 882–891.

[17] D. V. Arnold, An active-set evolution strategy for optimization with known constraints, in: International Conference on Parallel Problem Solving from Nature, Springer, 2016, pp. 192–202.

[18] D. V. Arnold, Reconsidering constraint release for active-set evolution strategies, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, ACM, New York, NY, USA, 2017, pp. 665–672. `doi:10.1145/3071178.3071294`.
URL `http://doi.acm.org/10.1145/3071178.3071294`

[19] I. Loshchilov, T. Glasmachers, H.-G. Beyer, Limited-memory matrix adaptation for large scale black-box optimization, CoRR abs/1705.06693 (2017).
URL `http://arxiv.org/abs/1705.06693`

[20] J. C. A. Barata, M. S. Hussein, The moore-penrose pseudoinverse: A tutorial review of the theory, Brazilian Journal of Physics 42 (1-2) (2012) 146–165. `doi:10.1007/s13538-011-0052-z`.

[21] R. Mallipeddi, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization, Tech. rep., Nanyang Technological University, Singapore (2010).

URL http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC10-Const/CEC10-Const.htm

[22] G. Wu, R. Mallipeddi, P. N. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization, Tech. rep., Nanyang Technological University, Singapore (2017). URL http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC2017/CEC2017.htm

[23] J. J. Liang, T. P. Runarsson, E. Mezura-Montes, M. Clerc, P. N. Suganthan, C. A. Coello Coello, K. Deb, Problem definitions and evaluation criteria for the CEC 2006 competition on constrained real-parameter optimization, Tech. rep., Nanyang Technological University, Singapore (2006). URL http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-06/CEC06.htm

[24] S. Finck, N. Hansen, R. Ros, A. Auger, COCO Documentation, Release 15.03 (2017).
URL http://coco.gforge.inria.fr/

[25] R. Tanabe, A. S. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in: IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 1658–1665. doi:10.1109/CEC.2014.6900380.

[26] A. Trivedi, K. Sanyal, P. Verma, D. Srinivasan, A unified differential evolution algorithm for constrained optimization problems, in: 2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017, 2017, pp. 1231–1238. doi:10.1109/CEC.2017.7969446.